Een eenvoudige pagina layout in HTML / CSS maken met een Flexbox

## inleiding

In deze opdracht maken we een eenvoudige webpagina waarbij we gebruik maken van een flexbox. Met behulp van de flexbox is het in HTML / CSS een stuk eenvoudiger om elementen netjes gestructureerd over je webpagina te verdelen. Je kunt de flexbox toepassen als container voor tabellen, callout boxes, afbeeldingen, buttons en nog veel meer. Het is zelfs mogelijk om je pagina layout met behulp van een flexbox te maken. Dat laatste gaan we in deze opdracht doen.

## De basis

We starten met een basis project. Maak deze aan in een aparte subfolder van je webroot locatie op je PC. Open je favoriete IDE om deze folder te openen en maak een nieuw *index.html* bestand. Maak tevens in een aparte subfolder die je *css* noemt een extern stylesheet bestand. Noem deze *styles.css*

Je structuur ziet er nu als volgt uit:

webroot

│ index.html

└───css

styles.css

Nu hebben we de basis van je project gelegd.

## De indexpagina

Open index.html en maak een standaard HTML pagina met de standaard elementen erin. Vraag je docent indien je niet weet hoe dit gaat.

Maak een link naar de het externe stylesheet bestand. Zet deze verwijzing in de <head> van je html document.

<link rel="stylesheet" href="css/styles.css">

Maak nu de volgende indeling in je <body> van je *index.html* document:

<nav>  
 navigation  
</nav>  
<main>  
<div class="container">  
 <div class="main-content">  
 Main Content  
 </div>  
 <aside class="related-links">  
 Related Links  
 </aside>  
 <aside class="ads">  
 Ads  
 </aside>  
</div>  
</main>  
<footer>  
 Footer  
</footer>

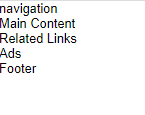
Zoals je ziet verdelen we de body in drie symantische elementen:

<nav>, <main> en <footer>

Binnen de <main> gebruiken we ook nog twee keer het symantische <aside> element. Deze symantische element bevatten niet zozeer speciale ingebouwde styling (behalve dat het block-level elementen zijn) maar dienen meer als hulp bij het beter indelen van je webpagina.

**Tip:** pagina’s welke gebruik maken van deze symantische onderdelen komen hoger in de ranking van search engines zoals Google of Bing

Bekijk nu de webpagina in je browser:



Zoals je ziet niets bijzonders, alles staat onder elkaar…

Geef aan waarom dit zo is:

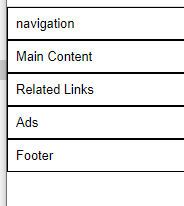
Antwoord:

## CSS styling toevoegen

Laten we nu wat styling toevoegen, open hiervoor het styles.css bestand.

Laten we starten met een border rondom de elementen te zetten om e.e.a. wat meer inzichtelijk te maken. Ook de ruimte vanaf de inhoud (content) tot deze border stellen we in:

nav, .main-content, .related-links, .ads, footer {  
 border: 1px solid black;  
 padding: 10px;  
  
}



Je ziet dat direct resultaat oplevert. De border strekt zich uit tot de breedte van je browserscherm en de ruimte rondom de content is 10 pixel groot geworden.

Uiteraard is dit nog steeds geen echte indeling waarmee je voor de dag kunt komen. Laten we dat aanpassen:

## Media query toevoegen

Met CSS is het mogelijk om bepaalde stukken CSS uit te voeren op basis van de breedte van de webbrowser. Er kan bijvoorbeeld herkent worden wat de breedte in pixels is van je webbrowser zodat bijvoorbeeld op een telefoon een andere style wordt gebruikt dan op je desktop computer. Dit kun je realiseren met zg. media-queries:  
  
Voeg de volgende media-query toe in je styles.css bestand:

@media (min-width: 400px) {

}

Wat hier wordt toegvoegd is een stukje CSS welke alleen wordt uitgevoerd wanneer de minimale breedte van je browser 400 pixels is. Indien de breedte lager is (dus kleiner dan 400 pixels) wordt dit stuk styling genegeerd.

Om nu te test of dit werkt kun je eevoudig even de achtergrond van je webpagina wijzigen:

@media (min-width: 400px) {  
 body {background-color: red; }  
}

Test dit uit in je browser. Wijzig de breedte van je window en zie wat er gebeurt.

**Tip:** open de inspector (ctrl-shift-i) om een mobiele telefoon te simuleren.

## Styling in mediaquery

Laten we nu de styling in de media-query iets meer laten doen wat we graag willen, een flexbox gebruiken. Pas de media-query aan:

@media (min-width: 400px) {  
   
 .container {  
 display: flex;  
 }  
  
 .main-content {  
 order: 2;  
 flex-grow: 3;  
 }  
  
 .related-links {  
 order: 1;  
 flex-grow: 1;  
 }  
  
 .ads {  
 order: 3;  
 flex-grow: 0.5;  
 }  
  
}

Je ziet dat de container class wordt voorzien van de display: flex eigenschap. Dit is wat een element een flexbox maakt. Alle onderliggende elementen volgen nu de eigenschappen van een flexbox.

Met een media-query kun je net als standaard CSS styling je selectors plaatsen binnen deze query. De geselcteerde elementen worden nu overschreven qua styling met de styling in deze query. Zou je dus een selector hebben gemaakt buiten deze media-query kun je in deze query deze waarde overschrijven indien er wordt voldaan aan de voorwaarde van deze media-query.

Laten we kijken naar de child elemeten van de flexbox:

Eén van de belangrijkste eigenschap van alle child elementen onder de flexbox is dat ze naast elkaar worden gezet in plaats van onder elkaar (zoals block-level elementen). Dit heeft de maken van de eigenschap: flex-direction. Deze staat standaard op flex-direction: row. Een rij gaat van links naar rechts, daarmee zou je het kunnen onthouden.

Kun je dit wijzigen? Jazeker. Pas de volgende code maar eens toe:

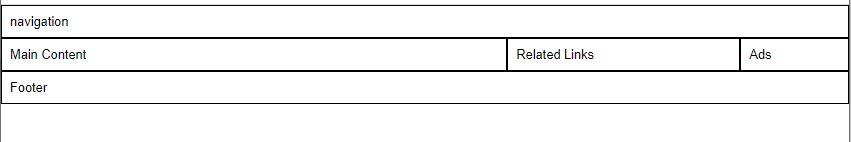
@media (min-width: 400px) {  
  
 .container {  
 display: flex;  
 **flex-direction: column;**  
 }  
  
 .main-content {  
 order: 2;  
 flex-grow: 3;  
 }  
  
 .related-links {  
 order: 1;  
 flex-grow: 1;  
 }  
  
 .ads {  
 order: 3;  
 flex-grow: 0.5;  
 }  
  
}

Als het goed is worden nu de elementen weer onder elkaar gezet.   
  
*flex-direction: column* zet dus de elementen onder elkaar (zoals in een kolom). De standaard is dus flex-direction: row. Om die rede kun deze eigenschap weglaten wanneer je element naast elkaar in een flexbox wilt zetten.

Kijk ook nog even naar de child elementen main-content, related-links en ads. Deze elementen zijn children van de flexbox en krijgen nu de mogelijkheid om bijvoorbeeld een andere volgorde aan te nemen dan standaard van links naar rechts in volgorde van code. Dit doe je met de order eigenschap. Speel hier maar eens mee en kijk naar het resultaat. Zie ook dit [artikel](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Ordering_Flex_Items) voor meer uitleg.

De flex-grow eigenschap is ook aardig om mee te experimenteren. Met deze eigeschap kun je de standaard breedte van een element in een flexbox wijzigen. Ik verwijs je naar het [volgende](https://developer.mozilla.org/en-US/docs/Web/CSS/flex-grow) artikel om hier meer over te weten te komen. Ga ook hier vooral mee experimenteren.

Het uiteindelijke resultaat zal er ongeveer als volgt uit moeten zien:



Zoals je ziet in het voorbeeld is de pagina nu ingedeeld in verschillende secties. Binnen deze secties kun je jouw content kwijt. Denk bijvoorbeeld aan een hoofdmenu in het navigation deel. De main content bevat informatie over het hoofdonderwerp. En wellicht plaats je leuke foto’s in het Ads gedeelte. De footer tenslotte kun je voorzien van een copyright infotext of iets dergelijk. Met andere woorden, je eigen fantasie is hier de enige beperking. Je mag de indeling en inhoud wijzigen naar gelang jou smaak en wensen.

## Gebruik maken van de schermhoogte

Laten we hier wat styling toevoegen om gebruik te maken van volledige hoogte van je browser. Standaard zal de browser de hoogte van de elementen uitrekken tot zover als nodig is. In dit geval dus 3 rijen met een tekst, border en padding van 10 pixel. Kortom een hoogte van circa 30 pixels totaal. Je browser is waarschijnlijk veel hoger. Laten we daar gebruik van maken.

Je totaal CSS moet er als volgt uitzien:

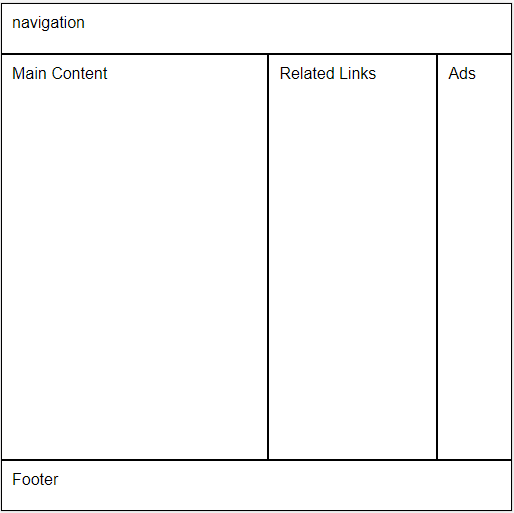
html,body {  
 box-sizing: border-box;  
 font-family: arial, helvetica, sans-serif;  
 margin: 0;  
 padding: 0;  
}  
  
img {  
 max-width: 100%;  
}  
  
\*,\*:before,\*:after {  
 box-sizing: inherit;  
}  
  
nav {  
 height: 10vh;  
}  
  
.container{  
 height: 80vh;  
}  
  
footer {  
 height: 10vh;  
}  
  
nav, .main-content, .related-links, .ads, footer {  
 border: 1px solid black;  
 padding: 10px;  
  
}  
  
@media (min-width: 400px) {  
 .container {  
 display: flex;  
 }  
  
 .main-content {  
 order: 1;  
 flex-grow: 3;  
 }  
  
 .related-links {  
 order: 2;  
 flex-grow: 1;  
 }  
  
 .ads {  
 order: 3;  
 flex-grow: 0.5;  
 }  
}

Zoals je ziet is dat de body, html selector bovenin de CSS de *box-sizing* eigenschap op *border-box* zet. Dit betekend dat te pixel calculaties iets makkelijk gaan werken. Zie het [volgende artikel](https://www.w3schools.com/css/css3_box-sizing.asp) voor meer uitleg hierover.

Verder zijn in de styling de hoogtes ingesteld van zowel de nav, container en footer element. Dit wordt gedaan door de *vh* (view-height) eenheid te gebruiken. 80vh staat voor 80% van de viewport hoogte. 10vh dus voor 10% van deze hoogte. Totaal is de hoogte van de viewport uiteraard 100vh (100%).

Een perfecte manier om de hoogte in te stellen naar gelang de hoogte van de browser hoogte.

Het resultaat zou er als volgt uit moeten zien:



Ga vooral het experiment aan door de browser in breedte maar ook in hoogte te wijzigen. Zie wat er gebeurd.