Inhoudsopgave

CoderDojo

Configuratie

I'm learning: HTML

1 We gebruiken Thimble voor het schrijven van de website code. We raden je aan om daar een account te maken. Het kost niets: alleen een e-mailadres is nodig. Ga naar https://thimble.mozilla.org/nl om je aan te melden.

2 Om de voorbeelden in deze Sushi kaarten te volgen, ga je naar dit project: https://thimbleprojects.org/cdkennemerwaard/392631 en klik op **Ombouwen**.
Linksboven zie je staan 'HTML/CSS Gevorderden 2 (remix)'. Verander als je wilt deze naam voor jezelf.

3 De voorbeelden in deze Sushi kaarten gaan allemaal om dit project, maar je kunt ook een website gebruiken die je zelf al eens gemaakt hebt.

4 Rechtsboven zie je 'voorbeeld' en 'auto' staan. Hieronder die je steeds het resultaat van je code. Het voorbeeld van je website wordt steeds automatisch aangepast als je de code wijzigt.

1 Met deze techniek kun je een diashow of fotogalerij maken, of een portfolio pagina waarop al je projecten te zien zijn: **preview cards** (= kaarten voorvertoning). Voeg de volgende HTML code aan je pagina toe:

```
<article class="card">
    <img src="paleisopdedam.jpg" class="smallPics">
    <h4>Paleis op de Dam</h4>
    <p>Amsterdam</p>
    </article>
```

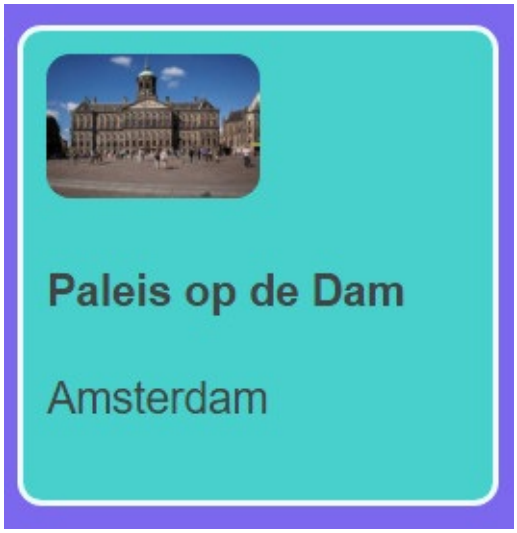
We gaan een paar kaarten maken met top bezienswaardigheden in Nederland. Je kunt zelf de afbeeldingen kiezen en tekst schrijven. Kies kaarten die bij de inhoud van jouw pagina passen.

2 Voeg deze CSS code toe voor de classes *card* en *smallPics* en de heading *h3*:

```
h3 {
  font-family: "League Gothic", sans-serif;
  font-style: normal;
  font-weight: 400;
}

.smallPics {
  height: 60px;
  border-radius: 10px;
}

.card {
  width: 200px;
  height: 200px;
  border: 2px solid #F0FFFF;
  border-radius: 10px;
  box-sizing: border-box;
  margin-top: 10px;
  font-family: "Lato", sans-serif;
}

.card:hover {
  border-color: #1E90FF;
}
```

De fonts (= lettertypes) League Gothic en Lato gebruiken we in veel CoderDojo materiaal.

3 Het zou er nu ongeveer zo uit moeten zien.We gaan er links van maken waarop mensen kunnen klikken om meer te zien.



4 Zet het hele **article** element in een link element. Let op: de eind `</a>` tag moet na de eind `</article>` tag staan!

```
<a href="Bezienswaardigheden.html#scPaleis">
<article class="card">
    <img src="paleisopdedam.jpg" class="smallPics">
    <h4>Paleis op de Dam</h4>
    <p>Amsterdam</p>
    </article>
      </a>
```

5 Zie je hoe de waarde van **href** eindigt op `#scPaleis` ? Dit is een trucje dat je kunt gebruiken om naar een specifiek gedeelte van een pagina te springen. Eerst type je de url van de pagina, gevolgd door **#**. In het codebestand van de pagina waar je heen linkt, zoek je het gedeelte waar je naar toe wilt springen en geef dat element een **id**, bijvoorbeeld < `section id="scPaleis">` . De waarde van de **id** is wat je na de **#** typt in je link.

6 Nu ziet de tekst er gek uit omdat het een link is. Dit kun je verbeteren door een **CSS class** toe te voegen aan de link, `class="cardLink"` .

```
.cardLink {
  color: inherit;
  text-decoration: none;
}
```

Als je de waarde van welke eigenschap ook zet op *inherit* (= overnemen), dan gebruikt het de waarde van het **parent** element (het bovenliggende element), dus nu komt de tekstkleur overeen met de tekst op de homepagina.

7 Maak nu zelf 4 of 5 van deze kaarten. Op de volgende Sushi kaart gaan we ze met een cool trucje op een goede plek op de webpagina zetten.

1 Ten eerste: alles in het midden (gecentreerd) zetten! Als je de linker- en rechtermarges op *auto* (automatisch) zet, kun je elk element in het midden zetten in plaats van helemaal links. Probeer het uit op de *.card* class.

```
margin-left: auto;
margin-right: auto;
```

2 Dat is de oplossing voor een veel voorkomend probleem. Een andere oplossing is alles op een rijtje zetten. Zet de card elementen die je gemaakt hebt in een nieuw container element. Het wordt geen **article** of **section**, maar eentje genaamd **div**. Het is een container voor algemeen gebruik die je kunt gebruiken om dingen te groeperen en mooi maken.

```
<div class="cardContainer">
```

3 Zet het volgende in je CSS stijlbestand:

```
.cardContainer {
  display: flex;
  flex-wrap: wrap;
  justify-content: space-around;
  padding: 10px;
}
```

Voilà! Door **Flex** staan je kaarten nu naast elkaar! Als je het voorbeeld schermvullend maakt, zie je hoe de plaats van de kaarten zich aanpast aan het scherm.

4 Verwijder de **width** (= breedte) en **height** (= hoogte) eigenschappen uit de *.card* class en kijk wat er gebeurt: **flex** past de kaarten in elkaar als bij een legpuzzel, en houdt de hoogte van alles wat er op een rijtje staat hetzelfde.

5 Je kunt dit trucje ook toepassen op het navigatiemenu boven aan je website. Zoek de CSS code voor je navigatiemenu. Verwijder `display: inline;` van de **list items**, dus uit het *nav ul li* blok. En type bij *nav ul* nu het volgende:

```
display: flex;
justify-content: flex-start;
```

Het menu ziet er nog hetzelfde uit, toch? Het mooie van flex is dat je de opmaak kunt bepalen met de eigenschap **justify-content** (= uitvullen van de inhoud). Verander de waarde daar in *flex-end* en kijk wat er gebeurt. Of verander het naar _space-around _om de menu items over de pagina te verdelen, net zoals bij de kaarten.

6 Een **responsieve** website is een website die zich aanpast aan de grootte van het scherm waarop je naar de website kijkt. Dus of je het op een pc, tablet of telefoon bekijkt: de website ziet er fantastisch uit. We gaan het menu responsief maken. Begin met de algemene stijlen: dit wordt je **default** (= standaard) gedrag.

7 Voeg de volgende CSS code toe aan je menu:

```
nav ul {
  padding: 0.5em;
  display: flex;
  flex-direction: column;
}

nav ul li {
  text-align: center;
  list-style-type: none;
  margin-right: 0.5em;
  margin-left: 0.5em;
}
```

8 Deze CSS code is voor mobiele apparaten: deze standaardstijl werkt goed voor smalle schermen. Zo maak je de stijlen voor grotere schermen:

```
@media all and (min-width: 600px) {
  nav ul {
    flex-direction: row;
    justify-content: space-around;
  }
}
```

Alles in dit blok is van toepassing wanneer het scherm breder is dan 600 pixels. Kun je een ander blok toevoegen voor schermen die groter zijn dan *800 pixels*, met *flex-end _in plaats van _space-around*?

* **Flex** is een behoorlijk sterk opmaakmiddel waarover we een hele Sushi serie zouden kunnen maken, maar hier kun je er meer over leren: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

9 Het voorbeeldscherm in Thimble is vrij klein. Rechts boven het voorbeeldscherm kun je op de knop met de vier pijltjes klikken om het op schermgrootte te zien.

Pas de breedte van het scherm een paar keer aan om het menu te zien veranderen. In deze blokken kun je alle CSS regels zetten die je maar wilt, om verschillende stijlen te maken voor verschillende groottes. Het zal nog van pas komen als je straks **css grids** opmaak gaat maken!

1 Als je een **caption** (= bijschrift) aan een afbeelding wilt toevoegen, een tekstje dat erbij hoort zoals een titel of korte beschrijving, dan kun je twee elementen gebruiken die speciaal daarvoor gemaakt zijn: **figure** (= figuur) en **figcaption** (= figuur bijschrift).

2 Zoek een **img** element op je website waarboven of onder tekst staat. Ik gebruik de afbeelding van Klara Koe, maar je kunt ook je eigen afbeeldingen gebruiken.

```
<img id="imgKlara" class="solidRoundBorders" src="Klara%20Koe.png"
alt="Klara de Koe">
<p>
  Jouw gids Klara!
</p>
```

3 Op de regel boven de code, zet je de tag `<figure></figure>` onder de code.

4 Haal nu de `<p></p>` tags weg en zet de tekst tussen `<figcaption></figcaption>` tags. Het zou er nu zo uit moeten zien:

```
<figure>
   <img id="imgKlara" class="solidRoundBorders" src="Klara%20Koe.png"
   alt="Klara de Koe">
  <figcaption>
  Jouw gids Klara!
  </figcaption>
</figure>
```

Het **figcaption** element is je **caption**. Het kan zowel onder als boven het **img** element geplaatst worden.

5 Als je de code uitvoert, kan het zijn dat de afbeelding en tekst van plaats zijn verandert. Misschien vond je de eerdere plaats beter en wil je dit helemaal niet. Je kunt zelf CSS regels maken voor **figure** en de marge op 0 zetten, of andere waarden geven die jij wilt. Je kunt stijlen maken voor **figure** en **figcaption** zoals voor elk ander element, met achtergrondkleuren, randen etc.

```
figure {
  margin-top: 0px;
  margin-bottom: 0px;
  margin-left: 0px;
  margin-right: 0px;
}
```

Het **figure** element gedraagt zich als een soort **container** van jouw afbeelding en bijschrift. Je kunt ze daardoor als één eenheid gebruiken wanneer je stijlen bepaalt, door ze bij elkaar te zetten behoud je ook een goede structuur op je website.

6 Een andere handige container is **aside** (= ernaast leggen). Je gebruikt het als je extra dingen hebt die niet echt horen bij de belangrijkste informatie op een pagina. Bijvoorbeeld: de Bezienswaardigheden pagina op mij website is een lijst met plaatsen om te bezoeken. Ik wil wat aantekeningen toevoegen over het weer en vervoer. Die informatie hoort niet echt bij het **article** element bij alle bezienswaardigheden.

7 Los van het article element, voeg je één of meer `<aside></aside>` tags toe met daarbinnen de extra informatie.

```
<aside class="lightPurpleBackground">
<h2>Rondreizen</h2>
<h3>Openbaar vervoer</h3>
<p>Je kunt heel goed met het openbaar vervoer reizen in Nederland.
Behalve met het openbaar vervoer kun je de bezienswaardigheden vaak ook
met een organisatie bezoeken.</p>
<h3>Auto</h3>
<p>Uiteraard kun je ervoor kiezen om met je eigen of een huurauto door
Nederland te toeren!</p>
</aside>
<aside class="lightPurpleBackground">
<h2>Het weer</h2>
  <p>Het weer in Nederland is aardig <span class="specialText">
  voorspelbaar</span>! Zorg altijd dat je een paraplu bij je hebt, en
  in de zomer zonnebrand!</p>
</aside>
```

* De **aside** en **article** containers zijn hetzelfde. Het enige verschil zit in de *betekenis*: dus waar gebruik je ze voor. Het is belangrijk om zinvolle html elementen te gebruiken. Het geeft je website een betere structuur en is met name handig voor mensen die je website op een **mobiel scherm** bekijken.

8 Je kunt CSS regels schrijven die **aside** elementen laten opvallen. Zag je het extra element **span**? Het is een extra tag die je kunt gebruiken om extra CSS toe te voegen! Je kunt van alles tussen **span** tags zetten. Het is makkelijk te gebruiken om een deel van een tekst of paragraaf aan te passen.

```
.lightPurpleBackground {
  background-color: #CFBFFF;
}
.specialText {
  color: #FF4500;
  font-size: larger;
}
```

9 Op de volgende kaart leer je hoe je de layout (= opmaak) interessanter kunt maken. Ter voorbereiding maak je een pagina met één **article** element en twee **aside** elementen binnen de `<main></main>` tags. Je mag ook verder gaan op de Bezienswaardigheden pagina.

1 Als het goed is, heb je een pagina waar drie elementen binnen **main** zitten: één **article** en twee **aside**. Voeg nieuwe CSS classes toe aan **main** met de drie elementen erin:

```
<main class="attPageLayoutGrid">
<article class="attGridArticle">
   <!--andere regels hier-->
</article>
<aside class="attGridAside1">
<!--andere regels hier-->
</aside>
<aside class="attGridAside1">
<!--andere regels hier-->
</aside>
</main>
```

Hiermee vervang je de opmaak van de **main** container, maar je kunt dit met elke soort container doen, zoals **div** of **article** en zelfs body. De techniek die je gebruikt heet **CSS grid** (= rooster).

2 Zet de **display** eigenschap van *grid* in de algemene container.

```
.altPageLayoutGrid {
  display: grid;
  grid-column-gap: 0.5em;
  grid-row-gap: 1em;
}
```

Wat denk je dat de **grid-column-gap** (= kolommen ruimte) en **grid-row-gap** (= rijen ruimte) eigenschappen doen?

3 Nu benoem je een **grid-area** (= gebied) voor elk element:

```
.attGridArticle {
  grid-area: agArticle;
}
.attGridAside1 {
  grid-area: agAside1;
}
.attGridAside2 {
  grid-area: agAside2;
}
```

4 En nu ga je je opmaak ontwerpen! We plaatsen de twee **aside** elementen naast elkaar onderaan. Hiervoor heb je twee **kolommen** van gelijke breedte nodig. Je kunt de **row** (= rij/regel) hoogte op automatisch laten staan. Zet de volgende code binnen de *.attPageLayoutGrid* CSS regels:

```
grid-template-rows: auto;
grid-template-columns: 1fr 1fr;
grid-template-areas:
  "agArticle agArticle"
  "agAside1 agAside2";
```

fr betekent *fraction* (= fractie, gedeelte). Je kunt *px* gebruiken voor exacte afmetingen in plaats van relatieve. Zie je hoe **article** de hele ruimte over twee kolommen benut?

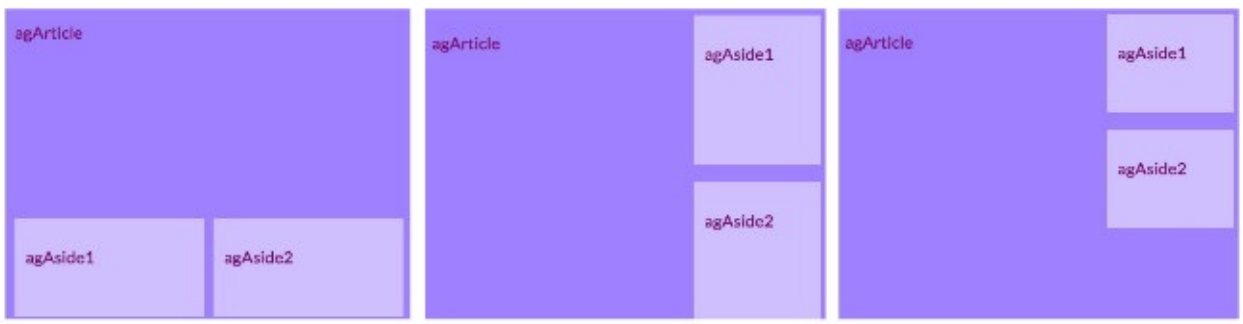
5 Probeer de **aside** elementen rechts te zetten, waarbij ze de helft van de breedte van **article** zijn. Verander de waarde van **grid-template-columns** en **grid-template-areas** naar:

```
grid-template-columns: 2fr 1fr;
grid-template-areas:
  "agArticle agArticle"
  "agAside1 agAside2";
```

6 Als je niet wilt dat de aside elementen helemaal naar beneden uitgerekt worden, kun je een blanco spatie gebruiken met behulp van een punt:

```
grid-template-areas:
  "agArticle agArticle"
  "agAside1 agAside2"
  "agArticle . ";
```

7 Zo ziet de opmaak eruit bij elk van de drie voorbeelden:



8 Met CSS grid kun je bijna elke opmaak maken die jij wilt. In dit voorbeeld hebben we header en footer eruit gelaten, maar ook deze kunnen binnen grid vallen. Als je meer over CSS grid wilt leren, kijk dan op https://css-tricks.com/snippets/css/complete-guide-grid/

1 Voeg een **div** element toe aan je pagina en zet er een paar afbeeldingen in. Geef de **div** en **img** elementen **id** waarden.

```
<div id="photoBox">
<img id="imgDeltawerken" src="delatwerken.jpg" alt="Deltawerken" />
<img id="imgVeluwe" src="veluwe.jpg" alt="Veluwe" />
</div>
```

Je zult CSS gebruiken om de foto's op de juiste plaats te zetten en een fotocollage te maken.

2 Voeg aan je CSS bestand regels toe voor elk element met behulp van **id selectors**.

```
#photoBox {
  width: 800px;
  height: 400px;
}
#imgDeltawerken {
  width: 120px;
}
#imgVeluwe {
  width: 250px;
}
```

3 De foto's komen één voor één tevoorschijn en houden de volgorde in je code aan. Om de exacte plaats te bepalen doe je twee dingen. Ten eerste voeg je de eigenschap `position: absolute;` toe aan de CSS regels van elke afbeelding. Ten tweede voeg je de eigenschap `position: relative;` toe aan de CSS regels van de container waarin ze zitten. Dit maakt het de **parent** van de afbeeldingen, dus wordt de plaats van de afbeeldingen daaraan gerelateerd.

4 Nu bepaal je de exacte plaats voor elke afbeelding. Je kunt vier eigenschappen gebruiken: **left** (= links), **right** (= rechts), **top** (= boven) en **bottom** (= onder). Deze bepalen hoe ver elke rand moet staan van de rand van de parent. Gebruik **top** of **bottom** voor de verticale positie en **left** of **right** voor de horizontale positie.

5 Deze code zet de Veluwe foto *100 pixels* vanaf de bovenkant en *60 pixels* vanaf de linkerkant.

```
#imgVeluwe {
  width: 250px;
  top: 100px;
  left: 60px;
  position: absolute;
}
```

* De waarden van de posities kunnen ook negatief zijn!

6 Je kunt ook bepalen of foto's over elkaar heen komen te staan, met behulp van de **z-index** eigenschap. De waarde daarvan kan elk heel getal zijn. De afbeelding met het *hoogste* getal komt bovenop te liggen!

```
#imgDeltawerken {
  width: 120px;
  top: 20px;
  left: 10px;
  position: absolute;
  z-index: 10;
}
#imgVeluwe {
  width: 250px;
  top: 100px;
  left: 60px;
  position: absolute;
  z-index: 7;
}
```

7 Voeg wat afbeeldingen toe en gebruik exacte posities met verschillende **z-index** waarden om een mooie collage te maken! Je kunt hiermee de positie voor alle html elementen bepalen, niet alleen voor afbeeldingen. Zet eens wat tekst over de foto's heen!

1 Laat de afbeeldingen bewegen als je er met de muis over heen gaat! Herinner je je nog de **transform** eigenschap van de HTML/CSS Gevorderden 1 reeks (toen je dingen liet draaien met **@keyframes** animaties)? Je kunt afbeeldingen iets omhoog en omlaag laten bewegen met *translateY* en *translateX*. Zoek de *.card:hover* CSS regels en verander ze naar het volgende. Probeer verschillende waarden uit in de translate (= vertaling) functie.

```
.card:hover {
  box-shadow: 0px 2px 2px rgba(0,0,0,0.2);
  transform: translateY(-2px);
}
```

Speel wat met verschillende pixelwaarden in de `box-shadow` eigenschap om te zien wat het doet.

2 *rgba(0,0,0,0.2)* is ook een manier om kleuren te bepalen. Het heeft meestal drie getallen (van 0 tot 255) voor Rood, Groen en Blauw. Het vierde getal, genaamd **alpha** waarde, bepaalt hoe doorzichtig iets is, het is een getal tussen **0** en **1**.

3 Maak de beweging tenslotte soepel door de volgende eigenschap toe te voegen aan *.card class*:

```
transition: all 0.2s ease-out;
```

Een duur van *0.2s* betekent dat de **transition** (= overgang) *0,2 seconden* duurt. In het Nederlands gebruiken we een komma bij dit soort getallen, in het Engels en ook in programmeertalen een punt.

4 Een ander effect dat je vast al eens bent tegengekomen op websites is **lightbox**: je klikt op iets waarna het scherm donkerder wordt, en iets anders (bijvoorbeeld een grotere afbeelding of popup) komt vooraan in je scherm te staan. Je maakt twee links voor dit effect.

5 De eerste link is voor de lightbox zelf. Het bevat alles dat tevoorschijn komt als je klikt. Vergeet niet om de link zelf een **id** te geven! In het voorbeeld zit het op de Bezienswaardigheden pagina van onze website. Doe dit bij welke pagina met afbeeldingen ook.

```
<a href="#_" class="lightbox" id="boxDuinen">
<h3>Duinen!!</h3>
<img src="Duinen.jpg" alt="Foto van duinen" class="bigPics"/>
<p>Mooi plaatje van de duinen in Nederland</p>
</a>
```

Je kunt van alles tussen de link tags zetten. Ik gebruik een grote afbeelding, een heading en wat tekst. Misschien wil jij alleen een afbeelding en geen tekst.

* Het maakt niet uit waar je deze code binnen het **main** element zet, je gaat het toch zo meteen onzichtbaar maken!

6 De andere link is natuurlijk datgene wat de lightbox tevoorschijn laat komen. Voeg een paar **a** tags toe aan het element, in dit geval een kleiner plaatje van het Waddengebied. Het **doel** van de link is de lightbox, welke je bepaalt hebt met **id**. Misschien herken je deze techniek van eerder.

```
<a href="#boxWaddengebied">
<img src="wadden.jpg" class="smallPics">
</a>
```

7 Dit is de CSS voor de lightbox. Kun je bedenken wat het doet?

```
.lightbox {
  background: rgba(0,0,0,0.8);
  color: #FFFFFF;
  text-align: center;
  text-decoration: none;
  width: 100%;
  height: 100%;
  top: 0;
  left: 0;
  position: fixed;
  visibility: hidden;
  z-index: 999;
}

.lightbox:target {
  visibility: visible;
}
```

Als je de **position** eigenschap op *fixed* (= vast) zet dan blijft het op z'n plek staan als je scrollt. De `:target` **pseudo-class** geldt alleen als de laatste link waarop geklikt is, de lightbox is. Dus als je op een andere plek klikt dan wordt de **visibility** (= zichtbaarheid) teruggezet naar **hidden** (= verstopt).

8 Je kunt zoveel lightboxes toevoegen als je zelf wilt. Ze kunnen allemaal dezelfde CSS class gebruiken. Wees er alleen zeker van dat elk een andere **id** heeft! Voor elke lightbox moet je van iets op je webpagina een link maken waarop je kunt klikken om de lightbox tevoorschijn te halen, en je gebruikt de **id** als de **href** waarde in die link; net zoals je hierboven hebt gedaan.