# CSS opmaak 2

### Voorbeeld

### Te gebruiken bij:

-

### Eigenschappen:

-  CSS Grid Layout in het leven geroepen: een tweedimensionaal raster-systeem.

Afbeelding met tekst, schermopname, ontwerp

Automatisch gegenereerde beschrijving

### Info

CSS is ongeveer net zo oud als de eerste webpagina's op het web. We hebben het altijd gebruikt om onze pagina's van styling te voorzien, maar het heeft nooit echt handig gewerkt als het op layouts aankwam. Eerst gebruikten we tabellen, daarna *floats*, toen kwam positionering en daarna inline-block.

Flexbox was hierin het eerste systeem dat echt *werkte*, en is een fantastische tool om layouts mee te bouwen. Het is echter nog steeds ééndimensionaal: items staan naast- of onder elkaar, maar kunnen niet overlappen. Daarom is **CSS Grid Layout** in het leven geroepen: een tweedimensionaal raster-systeem.

#### De display-property

Soms zal het echter zo zijn dat je bijvoorbeeld een duimpje rechts-onderin een foto wil plaatsen, zoals hier:

Afbeelding met schermopname, tekst, Graphics, grafische vormgeving

Automatisch gegenereerde beschrijving

hoe HTML-elementen zich van nature gedragen en hoe we dit kunnen manipuleren door middel van de display-property.

Alle HTML-elementen zijn te categoriseren als

* block element
  + Het element komt altijd op een nieuwe regel te staan ten opzichte van andere elementen;
  + Het element probeert altijd *alle* beschikbare breedte in zijn parent-element (container) te vullen. Wanneer hier ruimte voor is, betekent dit meestal dat het block element net zo breedt wordt als de container waar hij in staat;
  + width en height properties kunnen worden toegepast;
  + padding, margin en border zorgen ervoor dat andere elementen worden weggeduwd.
  + Tags zoals titels (<h1>, <h2>, etc) en <p>-tags gedragen zich van nature als block-elementen.
* inline element
  + Het element zal niet op een nieuwe regel komen te staan, maar op dezelfde regel als de content ervoor (tenzij het echt niet meer past!);
  + Het element wordt nooit groter dan zijn content;
  + width en height properties worden niet toegepast, ook al probeer je ze toch te gebruiken;
  + padding, margin en border zorgen ervoor dat andere elementen alleen horizontaal worden weggeduwd.
  + Elementen zoals <a>-tags en <span>-tags gedragen zich van nature als inline-elementen, maar met de display property kun je ieder element tot een inline-element maken.

Met CSS hebben we de mogelijkheid om deze properties te overschrijven of zelfs een derde variant te introduceren:

* inline-block
  + Het element zal niet op een nieuwe regel komen te staan, maar op dezelfde regel als de content ervoor tenzij het echt niet meer past;
  + width en height properties worden *wel* toegepast.
  + padding, margin en border zorgen ervoor dat andere elementen *alleen* *horizontaal* worden weggeduwd.

#### voorbeeld

.one {

display: inline;

background-color: blue;

}

.two {

display: block;

background-color: green;

}

.three {

display: inline-block;

background-color: red;

}

Afbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijving

#### beginnen

Laten we beginnen met de HTML. We hebben, wederom, een container-element nodig dat vier kleinere items omwikkelt:

<div>

<section class="item item-a">A</section>

<section class="item item-b">B</section>

<section class="item item-c">C</section>

<section class="item item-d">D</section>

</div>

Omdat we alle items dezelfde achtergrond-kleur en font-kleur willen geven, krijgen ze allemaal dezelfde "item" class. Daarnaast krijgen ze ieder een eigen class-naam, omdat we ze later ook nog individueel moeten kunnen aanspreken in onze grid. Laten we direct wat styling toevoegen om de boel overzichtelijk te houden:

.item {

background-color: #5B052B;

padding: 20px;

}

Onze *soon-to-be-grid* ziet er nu uit als één blok, omdat alle elementen grenzen aan elkaar:

Afbeelding met schermopname, Rechthoek, ontwerp

Automatisch gegenereerde beschrijving

Om te beginnen zullen we de container moeten omtoveren tot grid-container. Dit gaat verrassend makkelijk, met de display-property:

div {

display: grid;

}

Afbeelding met schermopname, tekst, ontwerp

Automatisch gegenereerde beschrijving

Het lijkt nu wellicht alsof al onze elementen er vandoor zijn gegaan, maar in werkelijkheid staan ze allemaal op een hoopje in de rechterbovenhoek. Dat komt omdat wij hebben gezegd dat ze vanaf nu in een raster staan, zonder te zeggen wáár ze dan moeten staan.

Dit doen we met de grid-area-property:

.item-a {

grid-area: a;

}

.item-b {

grid-area: b;

}

.item-c {

grid-area: c;

}

.item-d {

grid-area: d;

}

Voordat we ze kunnen toewijzen, moeten we het daadwerkelijke raster opstellen. Zoals je kon zien wilden we drie kolommen, en twee rijen. Laten we voor het gemak zeggen dat iedere kolom 140 pixels breed is, en iedere rij 140 pixels hoog. Om dit te declareren, gebruiken we de grid-template-columns en grid-template-rows property voor. De property grid-gap zal zorgen voor ruimte *tussen* de items.

div {

display: grid;

grid-template-columns: 140px 140px 140px;

grid-template-rows: 140px 140px;

grid-gap: 16px;

}

Afbeelding met roze, schermopname, Lila, Magenta

Automatisch gegenereerde beschrijving

Gelukkig is alles wat er nu gebeurt heel logisch. We hebben onze items zojuist een naam gegeven, beschreven hoeveel kolommen en hoeveel rijen we hebben en wat hun afmetingen zijn.

Maar we hebben nog steeds niet gezegd wáár de items in die kolommen en rijen thuishoren.

We zullen grid-area-benamingen van de items gebruiken om ze toe te wijzen in ons raster. Hiermee gaan we beschrijven over hoeveel rijen en kolommen het item moet vallen en wat de volgorde zal zijn.

In onze eerste rij moet *A* 2/3 van de rij in beslag nemen, en *B* 1/3. Op de tweede rij moet *C* 1/3, *D* 1/3 en *B* opnieuw 1/3 van de kolommen in beslag nemen:

div {

display: grid;

grid-template-columns: 140px 140px 140px;

grid-template-rows: 140px 140px;

grid-gap: 16px;

grid-template-areas:

"a a b"

"c d b";

}

Nu gedraagt het raster zich zoals we verwachten. Zie je hoe de aanhalingstekens iedere rij omwikkelen.

Afbeelding met tekst, Lettertype, roze, schermopname

Automatisch gegenereerde beschrijving

Dit zijn slechts enkele van de tientallen mogelijkheden van CSS Grid. ["The complete Guide to Grid"](https://css-tricks.com/snippets/css/complete-guide-grid/)

#### Vormen

CSS is in staat om allerlei vormen te maken.

elk element gedraagt zich als een box.   
Voeg een breedte en hoogte toe en je hebt precies de rechthoek die je nodig hebt.

##### *Alle* komen  vormen in CSS voort uit een vierkant.

* Voordat je een cirkel krijgt, maak je immers de randen van een vierkante <div>-tag rond.
* De daadwerkelijke breedte en hoogte van de driehoek worden bepaald door de dikte van de border.

div {

width: 0;

height: 0;

border-left: 5px solid transparent;

border-right: 5px solid transparent;

border-bottom: 5px solid black;

}

Dit geeft een driehoek waarvan de punt naar boven wijst. De

Afbeelding met tekst, schermopname, Rechthoek, Lettertype

Automatisch gegenereerde beschrijving

#### Pseudo-elementen

Een pseudo-element in CSS is een element dat eigenlijk niet bestaat, maar handmatig vóór of achter het hoofdelement geplakt kan worden.

a::after {...}

Een pseudo-element heeft altijd een *verplichte* property: content. Als je deze niet declareert, wordt het element niet weergegeven.   
Tevens kan het weglaten van de display property voor ongewenst gedrag zorgen.

a {

color: #D84282;

}

a::after {

content: "→";

display: inline-block;

}

We hebben in dit geval gekozen voor inline-block, omdat het link-element zich van nature als inline gedraagt en we willen dat het pijltje rechts van het hoofdelement komt te staan.   
In het geval van block komt het pijltje onder het hoofdelement te staan (block zorgt er immers voor dat alle omringende elementen naar de volgende regel worden geduwd).

##### voorbeeld

Je weet inmiddels hoe je een driehoek kunt maken. Laten we dit een stapje verder nemen en deze driehoek gebruiken om een praatwolkje te maken, zoals hier:

Afbeelding met Post-it-briefje, roze

Automatisch gegenereerde beschrijving

We hebben er twee nodig: een rechthoekige wolk en de uitstekende driehoek. Hiervoor gebruiken we **pseudo-elementen**.

Met een width, height, background-color en border-radius hebben we het grootste gedeelte al gedaan voor het vierkante wolkje. Nu willen we een driehoekje vóór de praatwolk plakken. Onthoud dat de properties display en content belangrijk zijn, anders gaan we onze driehoek niet zien:

div {

width: 120px;

height: 80px;

background-color: #D84282;

border-radius: 10px;

}

div::before {

content: "";

display: block;

width: 0;

height: 0;

border-top: 13px solid transparent;

border-right: 20px solid #D84282;

border-bottom: 13px solid transparent;

}

Zoals je ziet, hebben we de content-property een lege waarde toegewezen. Er hoeft immers geen content in onze driehoek komen te staan. **We moeten deze property echter wel altijd declareren.**

Tot nu toe gaat alles top, alleen ziet onze praatwolk er nu zo uit:

Afbeelding met schermopname, roze, Kleurrijkheid, Post-it-briefje

Automatisch gegenereerde beschrijving

Dit komt omdat pseudo-elementen altijd *binnen de box van hun echte element blijven*, tenzij wij dicteren dat ze ergens anders moeten staan. Dit doen we met de property position: absolute. Vergeet niet dat we hiervoor altijd het parent element een position: relative; moeten geven, anders zijn we straks onze driehoek kwijt:

div {

width: 120px;

height: 80px;

background-color: #D84282;

position: relative;

border-radius: 10px;

}

div::before {

content: "";

display: block;

width: 0;

height: 0;

border-top: 13px solid transparent;

border-right: 20px solid #D84282;

border-bottom: 13px solid transparent;

position: absolute;

left: -20px;

top: 26px;

}

En klaar is Kees

Afbeelding met Post-it-briefje, roze

Automatisch gegenereerde beschrijving

#### Transities

We maken hierbij onderscheid tussen twee concepten:

* **Transities**, waarbij de ene stijl vloeiend overgaat in de andere stijl (A → B).
* **Animaties**, waarbij er meerdere tussenstappen plaatsvinden om van begin naar eind te komen (A → B → C).

Afbeelding met tekst, Lettertype, logo, Graphics

Automatisch gegenereerde beschrijving

html

<a href="/">Naar de webshop</a>

Css

a,

a:link,

a:visited {

font-family: helvetica, sans-serif;

font-size: 30px;

color: #D84282;

text-decoration: none;

}

a::after {

content: "";

display: block;

width: 100%;

height: 3px;

background-color: #D84282;

}

Wanneer we een overgang willen creëren, is het belangrijk over de volgende dingen na te denken:

Hier hebben we de transition-property voor nodig! Deze property kan de volgende waardes toegewezen krijgen:

transition: [transition-property] [transition-duration] [transition-timing-function] [transition-delay]

Deze property kan de volgende waardes toegewezen krijgen:

transition: [transition-property] [transition-duration] [transition-timing-function] [transition-delay]

* *[transition-property]* - de property waarop we de transition willen toepassen (in dit geval width maar het kan ook opacity of all zijn).
* *[transition-duration]* - hoe lang de overgang moet duren (wij gaan 0.35 seconden gebruiken).
* *[transition-timing-function]* - de timing, waarbij we kunnen kiezen uit:

Afbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijving

* *[transition-delay]* (optioneel) - Mag de overgang direct beginnen of pas na een paar seconden? Als je deze waarde overslaat, begint de overgang altijd direct.

1. Wat is de beginstaat?
2. Wat is de eindstaat?
3. Hoe moet de overgang verlopen?
4. Laten we iets dieper ingaan op onze begin- en eindstaat. Omdat onze overgang de streep langzaam moet uitvullen, is onze beginstaat dus eigenlijk een streep met een *breedte van 0%* en de eindstaat een streep met een *breedte van 100%*. Laten we dat eens declareren.

Oké, we hebben de begin- en eindstaat vastgesteld,

/\* BEGINSTAAT \*/

a::after {

display: block;

content: "";

width: 0%;

height: 3px;

background-color: #D84282;

}

/\* EINDSTAAT \*/

a:hover::after {

/\* alle andere properties worden behouden, we overschrijven alléén de width property \*/

width: 100%;

}

a::after {

display: block;

content: "";

width: 0%;

height: 3px;

background-color: #D84282;

transition: width .35s ease-in-out;

}

Let op: de overgang van display: none naar een andere display property kun je niet animeren omdat het op het moment van display: none niet ‘bestaat’.