Welkom bij Learn CSS!

CSS INTRODUCTIE

(CSS)

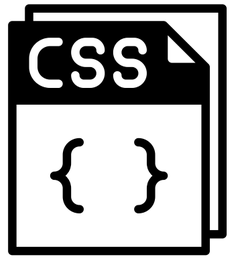
CASCADING STYLE SHEETS

- CSS is de taal die we gebruiken om een HTML-document te stylen.
- CSS beschrijft hoe HTML-elementen moeten worden weergegeven.
- Bijvoorbeeld om het lettertype, de kleur, de grootte en de spatiëring van de inhoud te wijzigen, deze in meerdere kolommen te splitsen of animaties en andere decoratieve elementen toe te voegen. Deze module biedt een overzicht van de basisprincipes hoe CSS werkt, hoe de syntax eruit ziet en hoe deze te gebruiken om styling aan uw HTML toe te voegen.

CSS KUNNEN WE OP DRIE MANIEREN TOEPASSEN OP ONZE HTML CODE.

- Inline CSS
- Style block in HTML
- Externe CSS

Externe CSS is de beste oplossing via de links hierboven kan je naar ieder onderdeel gaan.



INLINE-CSS

- Inline css heeft de hoogste prioriteit.
- we kunnen een stijl geven aan onze html tags door gebruik te maken van het style attribuut.

VOORBEELD:

```
<h1 style="color: red;">Title here</h1>
```

MEERDERE ELEMENT STIJLEN:

```
<h1 style="color: red; font-size: 10px;">Title here</h1>
```

NOTE:

Inline css is een techniek die niet echt wordt aangeraden. omdat deze het minst flexibel is en meer onderhoud en repetitief werk vergt.

Bekijk de online voorbeelden:

codepen.io

STYLE-BLOCK

Het style block is een html tag waarin we onze CSS kunnen definieren.

```
<style>
    <!-- css code here -->
</style>
```

```
<style>
    h1{
        color: red;
    }
</style>
```

STYLE BLOCKS

- Het style block moeten we plaatsen in de head tags.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Demo style block</title>

    <!-- style blocks here -->
    <style>
        h1{
            color: red;
        }
    </style>

</head>
<body>

    <header>
        <h1>Title Here</h1>
    </header>

</body>
</html>
```

CLASSES & IDENTIFIERS

Omdat we nu gebruik maken van het style block hebben we ook de mogelijkheid om **classes** en **identifiers** te gaan definieren binnen het style block.

- In HTML hebben we twee attributen class en id. de belangrijkste verschillen zijn een class mag meerdere keren voorkomen in een document en je kan meerdere classes toekennen aan een element, een id daarintegen moet uniek zijn en het id attribuut kan maar één referentie bevatten.

class:

```
<h1 class="c-red">Title here</h1>
<h1 class="c-red size-10">Title here</h1>
```

id:

```
<h1 id="c-red">Title here</h1>
```

Note:

wanneer je meerdere HTML elementen hebt met hetzelfde id zal de css wel werken over de vesrchillende elementen maar de javascript functie getElementById zal niet werken zoals je verwacht.

definiëren van classes en identifiers:

DEFINIËREN CLASS:

```
.c-red{
color: red;
}
```

Wanneer we een class definiëren plaatsen we voor de naam van onze class een (.).

DEFINIËREN ID (IDENTIFIER):

```
#myTitle{
        font-size:
20px;
}
```

Wanneer we een identifier definiëren plaatsen we voor de naam een (#).

Note:

Ook al is het style block een beter alternatief dan inline css dit is nog steeds niet ideaal we kunnen onze stijl nog niet delen onder verschillende pagina's.

HTML

```html
<h1 id="c-red">Title Here</h1>

<!— this is not working, multiple identifiers assigned —>
<h1 id="c-red size-10">Title Here</h1>
```

CSS

```css
#c-red{
    color: red;
}

#size-10{
  font-size: 10px;
}
```

HTML

```html
<h1 class="c-red">Title Here</h1>

<h1 class="c-red size-10">Title Here</h1>
```

CSS

```css
.c-red{
    color: red;
}

.size-10{
  font-size: 10px;
}
```

Bekijk de online voorbeelden:

codepen.io
jsfiddle.net
jsfiddle.net

EXTERNAL-CSS

In de twee voorgaande onderdelen hebben we bekeken hoe inline CSS en style block werkt en dat deze niet de beste oplossingen zijn voor de volgende redenen.

Nadelen inline css:

- niet onderhoudsvriendelijk
- altijd hoogste prioriteit dus minder flexibel
- veel herhaling
- maakt opmaak (HTML) minder leesbaar

Nadelen style block css:

- minder gebruiksvriendelijk dan external CSS
- HTML pagina specifiek

Eerst gaan we een extra folder maken **src** met daarin nog een folder genaamd css in deze folder plaatsen we al onze css bestanden die we dan gaan linken aan onze HTML pagina's.

Voorheen zag de structuur van ons project er als volgt uit.

CSS EN HTML LINKEN

- we hebben nu een extern css bestand in de volgende plaats "**src/css**" genaamd "**main.css**".

- om nu de link tussen onze HTML en CSS te maken gaan we gebruik maken van de tag **‹link›** deze plaatsen we ook weer in de **‹head›** van onze HTML pagina.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- link css -->
    <link rel="stylesheet" href="src/css/main.css">

    <title>Demo external css</title>
</head>
<body>

</body>
</html>
```

- **rel:** wordt gebruikt om de relatie tussen het huidige en het gekoppelde document op te geven.

- **href:** dit staat voor **"hyperlink reference"** en het identificeert een plaats (een hyperlink of gewoonlijk een link genoemd) in een hypertekst die de gebruiker naar een andere plaats linkt. Of een koppeling maakt met een ander bestand.

Link CSS via CDN

We hebben nu bekeken hoe we onze css in een apart bestand kunnen plaatsen, maar we kunnen ook naar css bestanden verwijzen die niet in ons project zitten.

```
<!-- CSS only -->
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-gH2yIJqKdNHPEq0n4Mqa/HGKIhSkIHeL5AyhkYV8i59U5AR6csBvApHHNl/vI1Bx" crossorigin="anonymous">
```

met deze link gebruiken we een css framework in dit geval gaat het over bootstrap. Om bootstrap te gebruiken kan je de source bestanden downloaden en toevoegen aan je project of via de **CDN** werken zoals het voorbeeld hierboven.

De code voor bootstrap te gebruiken met CDN kan je terugvinden op de website van Bootstrap(https://getbootstrap.com/) met de CDN kunnen we ook de javascript van bootstrap gebruiken.

OVERSCHRIJVEN VAN THIRDPARTY CSS CLASSES

Wanneer we een CSS-framework gebruiken (zoals Bootstrap), zijn er al standaard classes voorzien.

Als we deze stijlen willen aanpassen, doen we dit in ons **eigen CSS-bestand** door dezelfde class-namen te gebruiken.

Ons eigen CSS-bestand moet **na** het framework-CSS worden ingeladen, zodat onze regels voorrang krijgen.

Voorbeeld:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- link css -->
    <!-- framework css first -->
    <link rel="stylesheet" href="src/css/bootstrap.css">

    <!-- our main css -->
    <link rel="stylesheet" href="src/css/main.css">

    <title>Demo reset css</title>
</head>
<body>

</body>
</html>
```

CSS-SELECTORS

CSS-selectors worden gebruikt om **HTML-elementen** op een webpagina te selecteren en te stylen. We hebben ondertussen al de meest basic selectors bekeken: html tag, clas en id. We kunnen hier nog dieper op ingaan door combinaties te gaan maken. Met de volgende link(https://www.w3schools.com/cssref/trysel.php?) kan je de verschillen bekijken op een interactieve manier. Op de website van w3schools(https://www.w3schools.com/cssref/css_selectors.php) kan je een lijst terugvinden met alle selectors.

Hieronder is een lijst met selectors die regelmatig voorkomen.

Meest gebruikt!

Selecteer een class:

```
.class-name{
    // css here
}
```

Selecteer een id:

```
#id-name{
    // css here
}
```

Regelmatig gebruikt!
Selecteer meerdere HTML tags zoals alle divs en alle p tags:

```
div, p {
// css here
}
```

CSS-SELECTORS

Meest gebruikt!

Selecteer alle p tags in een div tag:

```
div p {
    // css here
}
```

Selecteer alle p tags waar de "parent" tag een div is:

```
div > p {
// css here
    }
```

Selecteer de eerste p tag binnen een div tag:

```
div + p {
// css here
    }
```

pseudo-classes

In CSS hebben we ook pseudo classes het is een kernwoord dat we kunnen toevoegen aan een selector. Bijvoorbeeld **:hover** is een pseudo class die heel vaak voorkomt in CSS. In deze cursus kan je de belangrijkste pseudo classes terugvinden er bestaan nog een hele reeks andere classes deze kan je terugvinden via deze link(https://www.w3schools.com/css/css_pseudo_classes.asp).

:hover Met deze pseudo class kunnen we eigenschappen aanpassen wanneer met onze muis over een element bewegen.

HTML

```html
<div class="example"></div>
```

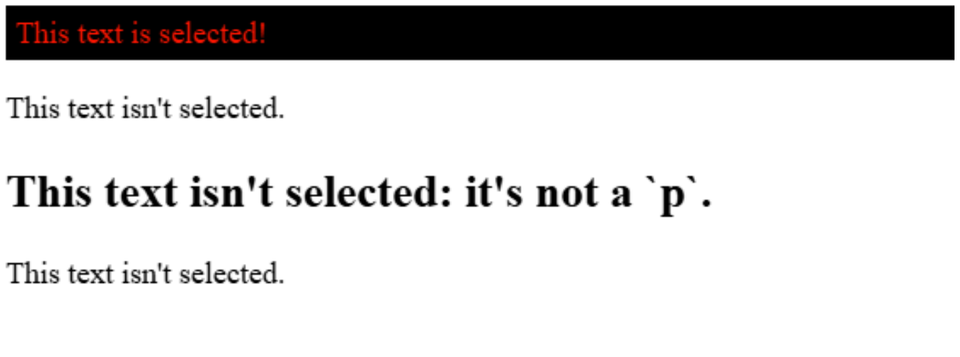
CSS

```css
.example {
    max-width: 100px;
    min-height: 50px;
    background-color: red;
}

.example:hover{
    background-color: blue;
}
```

pseudo-classes

:first-child De volgende pseudo class zal je de eerste child teruggeven van de eerste match dat het tegen komt.



HTML

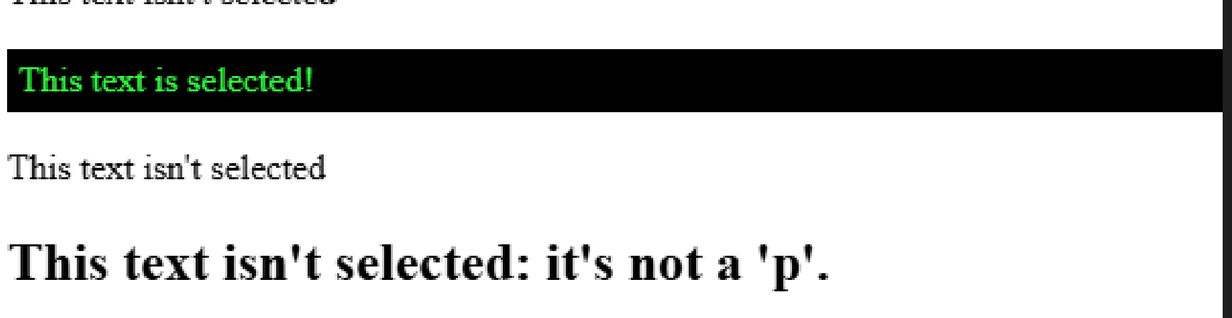
```
<div>
  <p>This text is selected!</p>
  <p>This text isn't selected.</p>
</div>
<div>
  <h2>This text isn't selected: it's not a `p`.</h2>
  <p>This text isn't selected.</p>
</div>
```

CSS

```
p:first-child {
  color: red;
  background-color: black;
  padding: 5px;
}
```

pseudo-classes

:last-child Natuurlijk hebben we ook de last-child als pseudo class dit zal het omgekeerde resultaat geven van :first-child.



HTML

```
<div>
  <p>This text isn't selected.</p>
  <p>This text is selected!</p>
</div>

<div>
  <p>This text isn't selected.</p>
  <h2>This text isn't selected: it's not a `p`.</h2>
</div>
```

CSS

```
p:last-child {
  color: lime;
  background-color: black;
  padding: 5px;
}
```

pseudo-classes

:nth-child() Met "nth-child" hebben we de mogelijkheid om een element(en) te gaan selecteren die niet eerst of laatst zijn. Tussen de haakjes kunnen we parameters meegeven. Deze pseudo class heeft ook **kernwoorden** zoals even en odd dit zal er voor zorgen dat je alle even of oneven elementen terugkrijgt.

item 1
item 2
item 3
item 4
item 5
item 1
item 2
item 3
item 4
item 5

HTML

```html
<div class="first">
    <p>item 1</p>
    <p>item 2</p>
    <p>item 3</p>
    <p>item 4</p>
    <p>item 5</p>
</div>

<div class="second">
    <p>item 1</p>
    <p>item 2</p>
    <p>item 3</p>
    <p>item 4</p>
    <p>item 5</p>
</div>
```

CSS

```css
.first p:nth-child(even){
    color: red;
}

.second p:nth-child(odd){
    color: blue;
}
```

pseudo-classes

Naast even of odd kunnen we ook een cijfer meegeven met de **:nth-child**.

Dit zal er voor zorgen dat om de drie items het derde item een blauwe kleur zal hebben.

item 1

item 2

item 3

item 4

item 5

HTML

```
<div>
    <p>item 1</p>
    <p>item 2</p>
    <p>item 3</p>
    <p>item 4</p>
    <p>item 5</p>
</div>
```

CSS

```
div p:nth-child(3n){
    color: blue;
}
```

Bekijk de online voorbeelden:
https://codepen.io/Quinten-Work/pen/jOxWmgr

CSS SPECIFICITY

In de vorige hoofstukken hebben we gezien dat er drie manieren zijn om **CSS** toe te passen op onze **HTML** in dit hoofdstuk gaan we ons verdiepen in CSS specificity.

Specificity is een algoritme dat door browsers wordt gebruikt om te bepalen welke css declaratie het meest relevant is voor een element, wat op zijn beurt zal bepalen hoe het element er zal gaan uitzien. Het specificiteitsalgoritme berekent het gewicht van de CSS selector. De selector met het grootste gewicht zal worden toegepast op het element.

Specificity hiërarchie:

- Inline css: inline css heeft de hoogste prioriteit.
- Identifiers (id): ID heeft de tweede hoogste prioriteit.
- Classes, Pseudo classes en attributen: Deze drie delen de derde plaats in de hiërarchie.
- Elementen en pseudo elementen: Deze hebben de laagste prioriteit.

Om het op een eenvoudigere manier te demonstreren heeft stuff & nonsense: https://stuffandnonsense.co.uk/archives/css_specificity_wars.html een overzicht gemaakt in de vorm van specificity wars.
Meer oefenen: https://codepen.io/Quinten-Work/pen/zYjrwXE

BOX-MODEL

In **CSS** wordt de term "**box-model**" gebruikt als het over ontwerp en lay-out gaat.

Het CSS box-model bepaalt hoe groot een element is en hoeveel ruimte het inneemt op de pagina. Het bestaat uit: **margin**, **border**, **padding** en de eigenlijke **content**. De afbeelding hieronder illustreert het boxmodel:

boxmodel

- **Margin:** Voegt een gebied toe buiten de border. De marge is transparant.
- **Border:** Een rand die rond de padding en content gaat.
- **Padding:** Voegt een gebied toe rond de content. De vulling is transparant.
- **Content:** De inhoud van het element, waar tekst en afbeeldingen verschijnen.

Note: Om te totale breedte of hoogte van een element te berekenen zijn we dus verplicht om de height of width van de content te nemen + padding + margin + border.

box-model voorbeeld

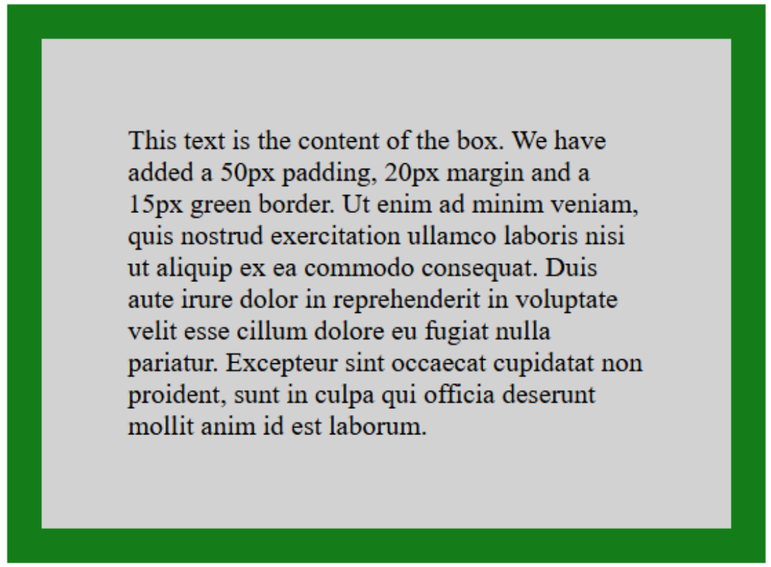
```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background-color: lightgrey;
  width: 300px;
  border: 20px solid green;
  padding: 50px;
  margin: 20px;
}
</style>
</head>
<body>

<h2>Demonstrating the Box Model</h2>

<p>The CSS box model is essentially a box that wraps around every HTML element.</p>
<p>It consists of (from innermost part to outermost part): content, padding, border, and
margin.</p>

<div>This text is the content of the box. We have added a 50px padding, 20px margin and
a 15px green border. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in
voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat
cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
</div>

</body>
</html>
```



Om te oefenen: https://www.w3schools.com/css/tryit.asp?filename=trycss_boxmodel

CSS-UNITS

In CSS hebben we verschillende soorten **units** om de lengte te gaan bepalen. Er zijn in CSS heel wat properties die een lengte aanvaarden als waarde zoals **width**, **height**, **margin**, **padding**, **font-size**, etc...

Lengte is een waarde gevolgt door een eenheid **px**, **em**, **cm**.

```
h1 {
font-size: 60px;
}

p {
font-size: 1.2em;
line-height: 50px;
}

div {
height: 3cm;
width: 45px;
}

article {
margin: 0;
}
```

In CSS bestaan er twee soorten lengtes **absolute** en **relative**.

Note: We mogen tussen het nummer en eenheid geen spatie laten!
Note: Wanneer we de waarde 0 meegeven mogen de eenheid weg laten!

Absolute lengte

De absolute lengte-eenheden zijn vast, een lengte in deze eenheid wordt uitgedrukt wordt precies weergegeven in deze maat.

Note: Pixels **px** is relatief ten opzichte van het apparaat, voor apparaten met een lage DPI zal **1px** gelijk zijn aan 1 pixel. Voor apparaten met een hoge DPI of printer zal **1px** meerdere pixels innemen.

Eenheid	beschrijving
cm	centimeters
mm	millimeters
in	inches (1in = 96px = 2.54cm)
px*	pixels (1px = 1/96th of 1in)
pt	points (1pt = 1/72 of 1in)
pc	picas (1pc = 12 pt)

Meer oefenen:

www.w3schools.com

Relatieve lengte

Relatieve lengte-eenheden specificeren een lengte ten opzichte van een andere lengte-eigenschap. Relatieve lengte-eenheden schalen beter tussen verschillende weergavemedia desktop en mobiele toestellen.

Met viewport bedoelen we het browser venster.

Note: De units **em** en **rem** zijn praktisch bij het maken van een schaalbare layout.

Meer oefenen:

https://www.w3schools.com/css/tryit.asp?filename=trycss_unit_rem

Eenheid	beschrijving
em	Ten opzichte van de lettergrootte van het element (2em betekent 2 keer de grootte van het huidige lettertype)
ex	Ten opzichte van de x-hoogte van het huidige lettertype (zelden gebruikt)
ch	Ten opzichte van de breedte van het cijfer "0" (nul)
rem	Ten opzichte van de lettergrootte van het hoofdelement
vw	Ten opzichte van 1% van de breedte van de viewport*
vh	Ten opzichte van 1% van de hoogte van de viewport*
vmin	Ten opzichte van 1% van de hoogte of breedte van de viewport* afhankelijk van welke het kleinst is.
vmax	Ten opzichte van 1% van de hoogte of breedte van de viewport* afhankelijk van welke het grootst is.
%	Ten opzichte van het bovenliggende element

voorbeeld voor rem Unit:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
 font-size:16px;
}

div {
  font-size: 2rem;
  border: 1px solid black;
}

</style>
</head>
<body>

<p>The font-size of this document is 16px.</p>

<div>The font-size of this div element is 2rem.</div>

<p>The rem unit sets the font-size relative to the browsers base font-size, and will not
inherit from its parents.</p>

</body>
</html>
```

The font-size of this document is 16px.

The font-size of this div element is 2rem.

The rem unit sets the font-size relative to the browsers base font-size, and will not inherit from its parents.

OVERFLOW

Met de CSS **overflow** property kunnen we gaan bepalen wat er met content moet gebeuren wanneer deze groter is dan de oppervlakte.

De overflow property heeft verschillende waardes die kunnen gebruikt worden:

- **visible:** Dit is de standaard waarde. De content is niet afgekapt. De inhoud wordt buiten het vak van het element weergegeven.
- **hidden:** De content wordt afgekapt en de rest van de inhoud is onzichtbaar.
- **scroll:** De content wordt afgekapt en er wordt een scrollbar toegevoegd om de rest van de inhoud te zien.
- **auto:** Vergelijkbaar met scrollen, maar het voegt alleen scrollbars toe als dat nodig is.

Note: De waarde **overflow** werkt enkel voor blokelementen die en specifieke hoogte hebben.

Note: Bij OS X lion (apple mac) zijn scollbars standaard verborgen enkel wanneer het element gebruikt wordt zal de scrollbar zichtbaar worden (ook wanneer de overflow waarde scroll is).

```html
<!DOCTYPE html>
<html>
<head>
<style>
#overflowTest {
  background: #04AA6D;
  color: white;
  padding: 15px;
  width: 90%;
  height: 100px;
  overflow: scroll;
  border: 1px solid #ccc;
}
</style>
</head>
<body>

<h2>CSS overflow Property</h2>

<p>The overflow property controls what happens to content that is too big to fit into an area.</p>

<p>Here, scrollbars are added on overflow:</p>

<div id="overflowTest">Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem.</div>

</body>
</html>
```

CSS overflow Property

The overflow property controls what happens to content that is too big to fit into an area.

Here, scrollbars are added on overflow:

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent

visible

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Fuga totam atque, deserunt, alias tempora officia, laboriosam explicabo veniam dolores possimus cumque impedit rerum? Distinctio eum necessitatibus delectus et non accusamus. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Fuga totam atque, deserunt, alias tempora officia, laboriosam explicabo veniam dolores possimus cumque impedit rerum? Distinctio eum necessitatibus delectus et non accusamus. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Fuga totam atque, deserunt,

hidden

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Fuga totam atque, deserunt, alias tempora officia, laboriosam explicabo veniam

Scroll

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Fuga totam atque, deserunt, alias

auto

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Fuga totam atque, deserunt, alias tempora officia, laboriosam

Bekijk de online voorbeelden:

codepen.io
www.w3schools.com

FLOAT-AND-CLEAR

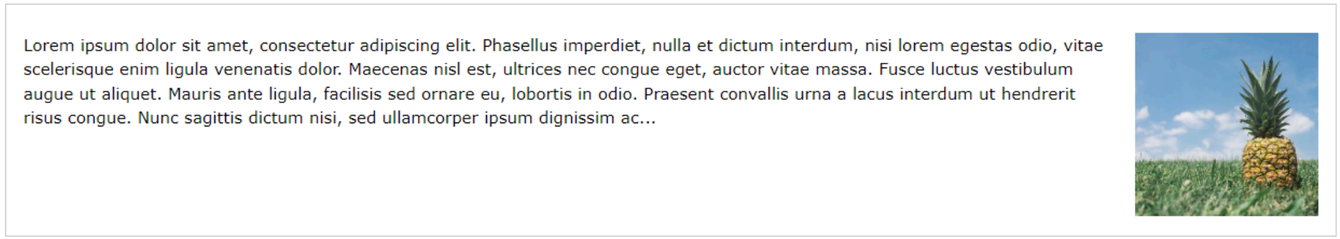
Float

Met de CSS **float** property kunnen we bepalen hoe een element moet zweven. Er zijn verschillende waardes die de float property accepteert.
De clear property zal aanduiden welke elementen kunnen zweven naast een cleared element en aan welke kant.

- **left:** Met de waarde **left** zal het element links zweven in de container.
- **right:** Met de waarde **right** zal het element rechts zweven in de container.
- **none:** Bij deze waarde zal het element niet zweven het zal worden getoond waar het
- **inherit:** Een element met deze waarde erft de **float** waarde van zijn parent

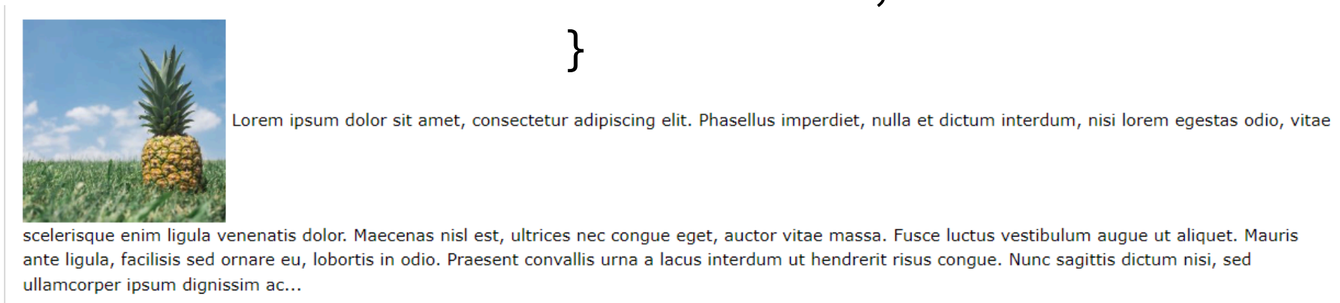
Voorbeeld:

```
img {
    float: right;
}
```



Voorbeeld:

```
img {
    float: none;
}
```



CLEAR

Wanneer we gebruik maken van float maar we willen het volgende element niet naast het zwevende element hebben maar eronder, gebruiken we de clear property.

Met de clear property kunnen we verschillende waardes meegeven.

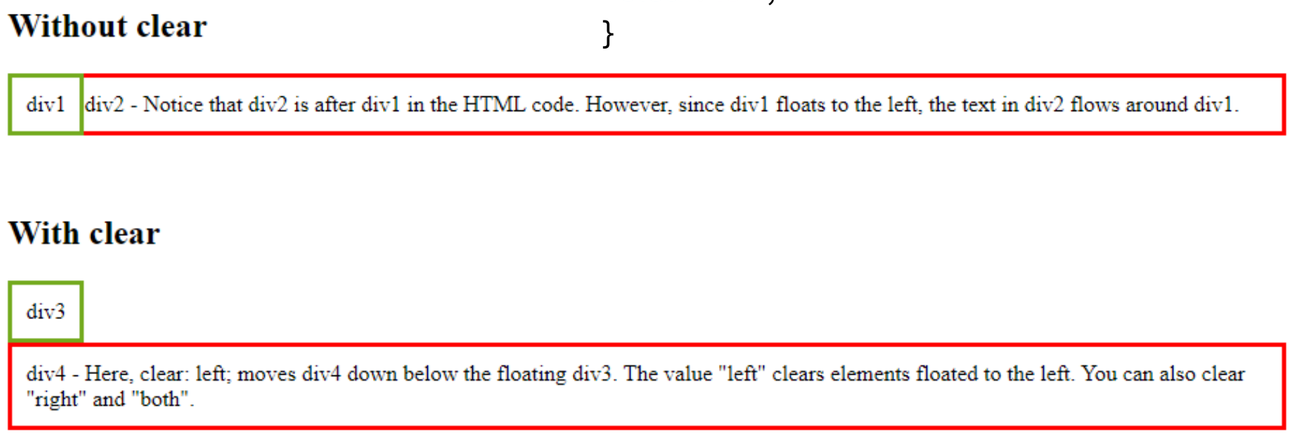
- **none:** Het element zal niet naar onder, links of rechts geduwd worden.
- **left:** Het element wordt onder links zwevende elementen geduwd.
- **right:** Het element wordt onder rechts zwevende elementen geduwd.
- **both:** Het element wordt onder links en rechts zwevende elementen geduwd.
- **inherit:** Het element zal de waarde overnemen van zijn parent/bovenliggend element.

```
.div1 {
  float: left;
  padding: 10px;
  border: 3px solid #73AD21;
}

.div2 {
  padding: 10px;
  border: 3px solid red;
}

.div3 {
  float: left;
  padding: 10px;
  border: 3px solid #73AD21;
}

.div4 {
  padding: 10px;
  border: 3px solid red;
  clear: left;
}
```



Z-INDEX

Tot nu toe hebben altijd elementen verplaatst op de **'X'** en **'Y'** as. Met de **z-index** property kunnen we bepalen welk element boven of onder een ander element ligt. Op deze manier kunnen elementen elkaar overlappen.

Note: de **z-index** property werkt alleen bij elementen die een **position** hebben (**relative, absolute, fixed of sticky**).

Note: Wanneer elementen dezelde **z-index** hebben, wordt het element dat later in de HTML-code staat bovenaan weergegeven.

voorbeeld:

```
img {
  position: absolute;
  left: 0px;
  top: 0px;
  z-index: -1;
}
```

This is a heading

Because the image has a z-index of -1, it will be placed behind the text.

Bekijk de online voorbeelden:

codepen.io
www.w3schools.com

CSS-VARIABLES

CSS variabelen (ook **Custom properties** genoemd) zijn entiteiten die we kunnen definieren in ons CSS bestand het heeft ons de mogelijkheid om waardes te kunnen hergebruiken in meerdere CSS classes. Grotere websites hebben een hele reeks aan CSS classes vaak met dezelfde waarde. Bijvoorbeeld een kleur dat op tientallen plaatsen gebruikt wordt wanneer we deze zouden willen aanpassen moeten we alle plaatsen opzoeken om dit dan manueel aan te passen. Door gebruik te maken van variabelen kunnen we dit probleem gaan vermijden.

Variabelen declareren

Een variabele kunnen we declareren door twee streepjes **(--)** te gebruiken gevolgd door de naam van onze variabele. Variabelen kunnen we declareren in de **global scope** of de **local scope**.

voorbeeld:

```
:root{
  --blue: #1e90ff;
  --white: #ffffff;
}
```

Bekijk de online voorbeelden:
www.w3schools.com

global scope

In onderstaand voorbeeld declareren we twee variabelen in de global scope genaams blue en white, beide variabelen bevatten een rgb waarde dat hun kleur voorsteld.

```
:root{
  --blue: #1e90ff;
  --white: #ffffff;
}
```

Deze variabelen kunnen we nu hergebruiken in andere css classes of identifiers met de **var()** functie.

```
h1{
    background-color: var(--white);
    color: var(--blue);
}

.container{
    background-color: var(--blue);
}

button{
    background-color: var(--white);
    color: black;
    border-color: var(--blue);
}
```

Vervolgens gebruiken we de **var()**-functie om de waarde van de variabelen in het stijlblad in te voegen.

local variables

Lokale variabelen gaan we direct in de scope declareren van een selector.

```
h2{
   --red: #ff0000;
   background-color: var(--white);
   color: var(--red);
}
```

override variables

We hebben ook nog de mogelijkheid om variabelen te overschrijven. Soms willen we namelijk voor een bepaalde sectie een andere waarde meegeven.

```
article h1{
    --blue: #3366ff;
    background-color: var(--white);
    color: var(--blue);
}
```

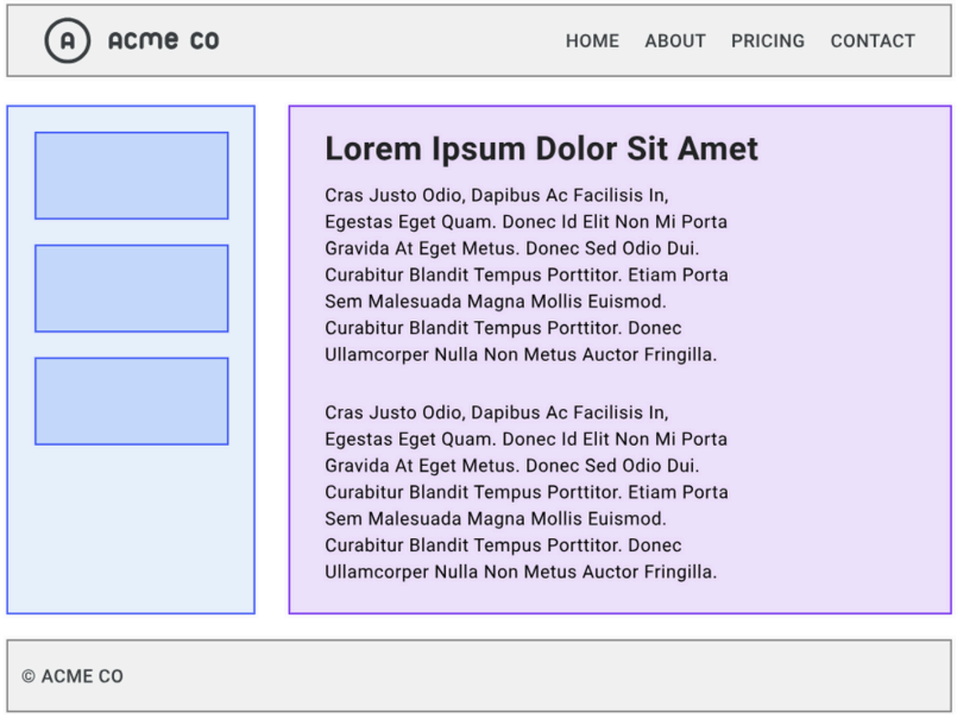
Bekijk de online voorbeelden:
codepen.io

CSS-GRID

CSS grid is een layout module die ingebouwd is in CSS, het is een grid gebaseerd layout systeem. Het bestaat uit rijen en kolommen wat het eenvoudiger maakt om een web pagina te ontwerpen zonder dat we ons zorgen moeten maken over float en positionering van elementen.
Het CSS grid layout systeem bestaat uit een container als parent element met één of meerdere child elementen.

Om een grid layout samen te stellen moeten we eerst een parent element declareren die we gaan gebruiken als container.

```
.grid-container{
  display: grid;
}
```



Meer kennis over grid:
web.dev

CSS-grid

Om een **grid** layout samen te stellen moeten we eerst een parent element declareren die we gaan gebruiken als container.

```
.grid-container{
    display: grid;
}
```

De volgende stap is het bepalen van het aantal kolommen en rijen dit wordt beschreven als grid template areas.

```
.grid-container{
    display: grid;
    grid-template-areas:
    'header   header  header  header  header  header'
    'menu     main    main    main    right   right'
    'menu     footer  footer  footer  footer  footer';
}
```

TIP: Plaats tabs tussen de verschillende woorden om een duidelijker overzicht te krijgen van de verschillende kolommen en rijen.

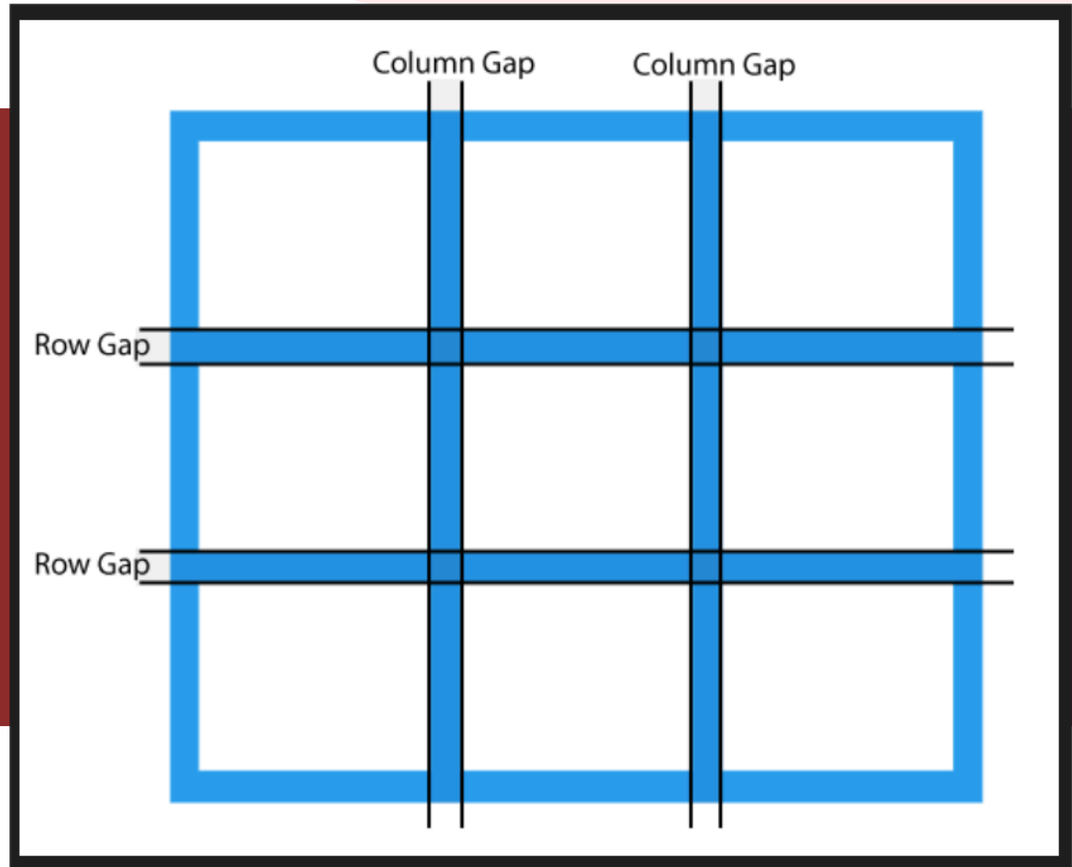
Nu hebben we verschillende areas aangemaakt **header**, **menu**, **main** en **footer**.

CSS-grid

Tussen de verschillende rijen en kolommen kunnen we ook een ruimte voorzien. **column-gap**, **row-gap** en **gap**.

```
.grid-container {
  display: grid;
  gap: 50px;
}
```

Naast grid is er ook nog de **inline-grid** variant, wanneer we gaan voor grid zal de flow van het voorgaande element en het volgende element onderbroken worden. Bij inline-grid zal de container zich aanpassen naar de flow van het voorgaande en het volgende element.

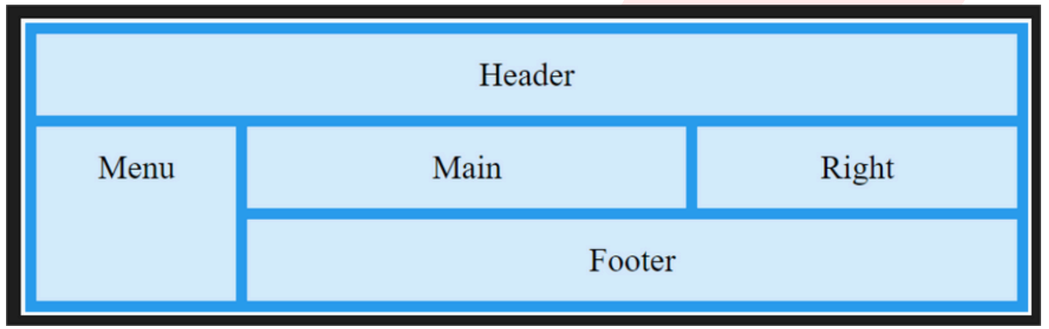


online example:

www.js-craft.io

CSS-grid

Onderstaand voorbeeld bestaat uit zes kolommen en drie rijen. En maakt gebruik van **display: grid;**



De HTML zal er als volgt uitzien.

```html
<div class="grid-container">
  <div class="my-header">Header</div>
  <div class="my-menu">Menu</div>
  <div class="my-main">Main</div>
  <div class="my-right-menu">Right</div>
  <div class="my-footer">Footer</div>
</div>
```

Bekijk de online voorbeelden:

codepen.io

CSS

Wanneer we onze layout hebben gedefinieerd kunnen we elementen plaatsen in hun area. Dit doen we door een class te maken in CSS en deze de grid-area eigenschap mee te geven.

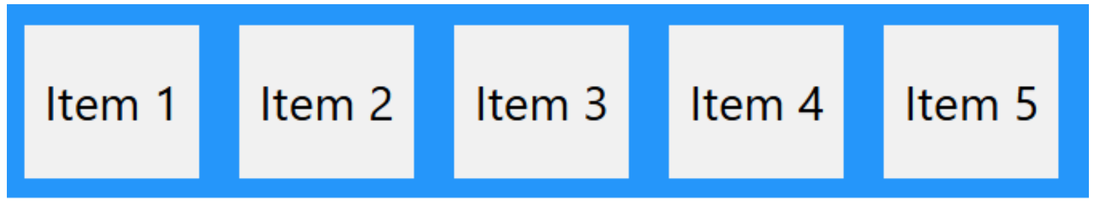
```css
.my-header { grid-area: header; }
.my-menu { grid-area: menu; }
.my-main { grid-area: main; }
.my-right-menu { grid-area: right; }
.my-footer { grid-area: footer; }

.grid-container {
  display: grid;
  grid-template-areas:
    'header  header  header  header  header  header'
    'menu    main    main    main    right   right'
    'menu    footer  footer  footer  footer  footer';
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}

.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
```

CSS-FLEXBOX

CSS Flexbox is een layoutmodule die ingebouwd is in CSS. Het is een flexibel layoutsysteem dat vooral geschikt is voor **mobile-first webdesign**. Met Flexbox kunnen we elementen eenvoudig uitlijnen en verdelen, zonder gebruik te maken van float of complexe positionering.



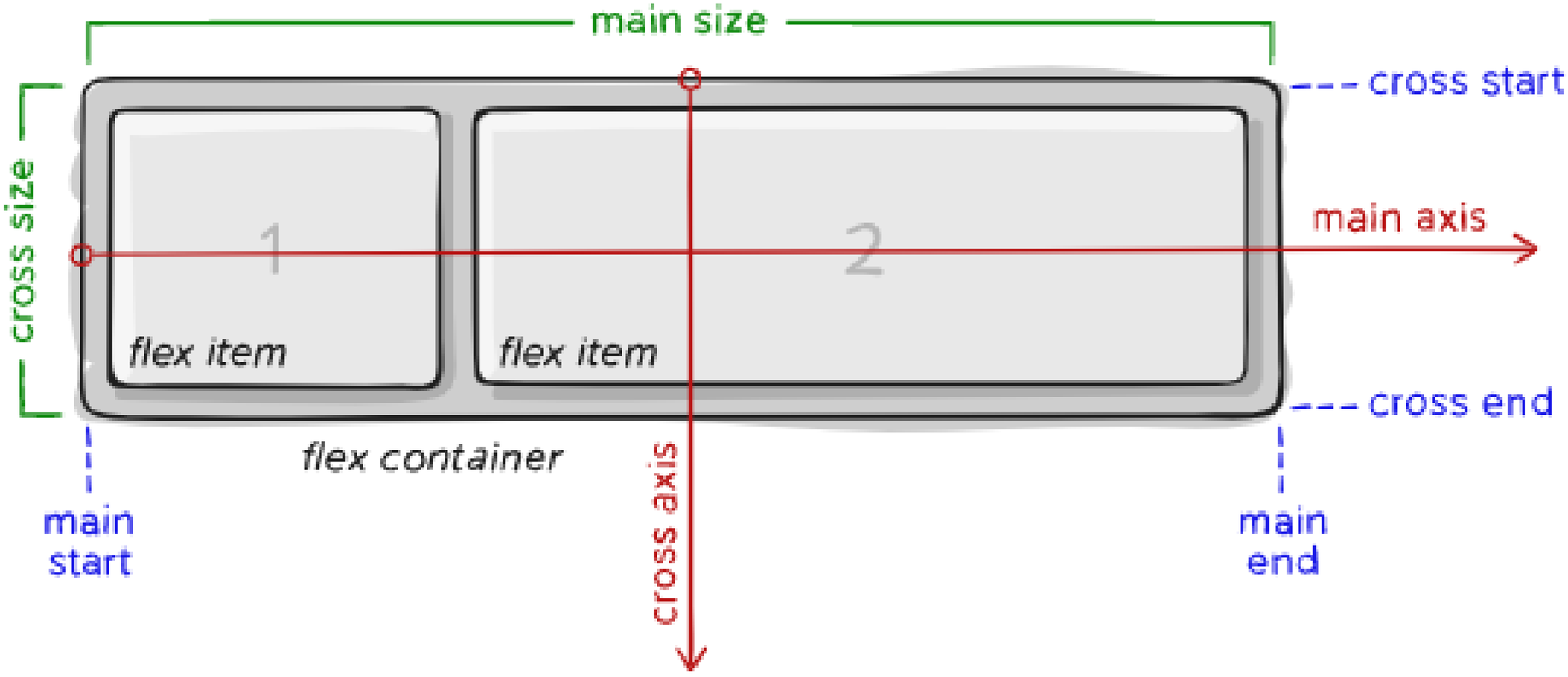
Terminologie

Wanneer we gebruik maken van CSS flexbox moeten we eerst een aantal belangrijke principes gaan bekijken.
de **main axis** van (main-start tot main-end) of de **cross axis** van (cross-start tot cross-end).

- **main axis**: De hoofdrichting waarin de flex-items worden geplaatst. Deze wordt bepaald in CSS door de **flex-direction** property.
- **main-start** en **main-end**: De start- en eindpositie van de main axis. Flex-items worden geplaatst van main-start naar main-end.
- **main size**: Dit is de breedte en de hoogte van een flex-container.
- **cross axis**: De cross axis is de as die kruist met de main axis wanneer de main axis verticaal loopt zal de cross axis horizontaal lopen. Wanneer de main axis horizontaal loopt zal de cross axis verticaal lopen op deze manier krijgen we altijd een kruis vandaar cross axis.
- **cross-start** | **cross-end**: Flexlijnen worden gevuld met items en in de container geplaatst, beginnend aan de cross-start van de flexcontainer en gaand naar de cross-end.
- **cross-size**: De breedte of hoogte van de flex-container of de kruisende as de breedte of de hoogte is afhankelijk van de items in de flex-container.

flex-terminology

Elementen volgens telkens de **main axis** van (main-start tot main-end) of de **cross axis** van (cross-start tot cross-end).



Meer lezen:
web.dev

flex container

Om gebruik te kunnen maken van flexbox moeten we een flex container aanmaken. Gebruik hiervoor de display property met de waarde flex.

CSS

```
.flex-container{
display: flex;
}
```

Wanneer we een flex container hebben gedefinieerd zal de standaard richting van de elementen een rij zijn.

flex row

Met de property **flex-direction** kunnen we de richting van onze elementen bepalen **row** of **column**.

flex column

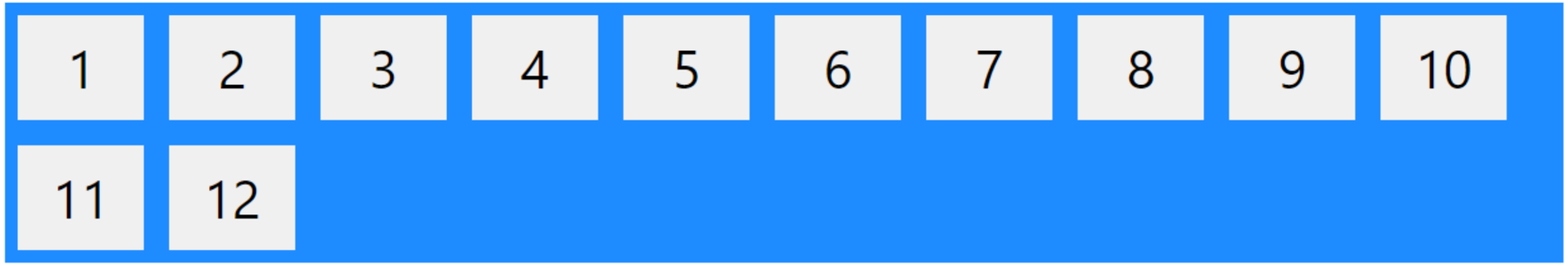
flex properties

CSS

```
.flex-container {
  display: flex;
  flex-wrap: wrap;
}
```

Wanneer elementen buiten de container dreigen te vallen, kunnen we dit opvangen met de **flex-wrap** property.

Wanneer we een flex container hebben gedefinieerd zal de standaard richting van de elementen een rij zijn.



flex wrap

Items kunnen ook worden uitgelijnd dit doen we met de justify-content property

justify content

```
.flex-container {
  display: flex;
  justify-content: center;
}
```

Note: De justify content property is van toepassing op de **main axis**

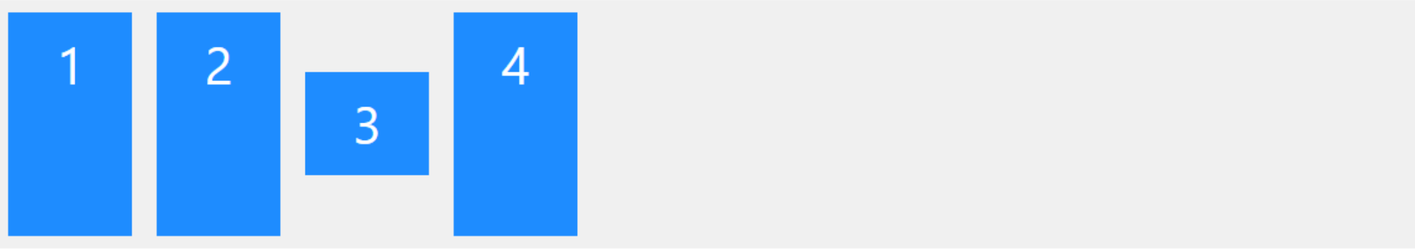
flex properties

Naast **center** zijn er ook nog de waardes **flex-start**, **flex-end**, **space-around** en **space-between**.

Om elementen uit te lijnen op de **cross axis** kunnen we gebruik maken van de property **align items**.

Met de **align-self** property is het mogelijk om individuele elementen uit te lijnen op de **cross axis**.

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="align-self: center">3</div>
  <div>4</div>
</div>
```



align self

Bekijk de online voorbeelden:

css-tricks.com

THANK YOU