DOCVARIABLE

Essentials

**Auteur:** Johan Cleuren

**Lectoren:** Johan Cleuren
Tim Dupont
Cedriek Vos
Reinaut Krekels
Carine Derkoningen

Nele Custers

# 1 Inhoudsopgave

# 2 Positioneren met CSS

## 3 Waarom CSS positioning

Vanaf de begindagen van het web ervaren ontwikkelaars de nood om een webpagina in te delen in deelvensters. Lang werd dit met frames gedaan. Dit wordt momenteel als een achterhaalde en “old school” techniek beschouwd. Het gebruik van tabellen als alternatief heeft ook zijn nadelen: het vereist zeer veel HTML-code met een slechtere leesbaarheid tot gevolg. Door de grote hoeveelheid en de relatieve complexheid duurt het ook langer om de pagina met tabellen op te bouwen. Tabellen schenken ook geen absolute vrijheid aangezien alles in een rechthoekige opmaak moet passen en verstoren de semantische opmaak.

Een ander alternatief is om gebruik te maken van layers die zelfs boven elkaar kunnen geplaatst worden. De term layer wordt gebruikt voor elementen die gebruik maken van positioned lay-out, een techniek om elementen te positioneren zonder veel interactie van/met andere elementen. Dit geeft een haast onbegrensde vrijheid, aangezien je deze layers niet alleen exact kan plaatsen, je kan ook alle kenmerken zoals bijvoorbeeld breedte, kleur of zichtbaarheid kan bepalen. Het meest gebruikte voorbeeld is de multikolommenopmaak die toelaat om een pagina er te laten uitzien als een magazine of krant.

De volgende CSS-attributen worden hierbij het meest gebruikt:

| CSS | Waarde | Werking |
|---|---|---|
| bottom | auto<br>%<br>afstand | De afstand vanaf de benedenrand van de *nearest positioned ancestor* of van het browservenster. |
| clear | left<br>right<br>both<br>none | Bepaalt de grenzen van floating elementen. |
| clip | auto<br>vorm | Bepaalt de vorm van het element. |
| float | left<br>right<br>none | Bepaalt de positie van het element in haar parent element. |
| left | auto<br>%<br>afstand | De afstand vanaf de linkerrand van de *nearest positioned ancestor* of van het browservenster. |
| overflow | auto<br>hidden<br>scroll<br>visible | Bepaalt wat moet gebeuren als de inhoud groter wordt dan het beschikbare oppervlak. |
| position | static<br>relative<br>absolute<br>fixed | Plaatst het element op een statische, relatieve, absolute of vaste positie. |
| right | auto<br>%<br>afstand | De afstand vanaf de rechterrand van de *nearest positioned ancestor* of van het browservenster. |
| top | auto<br>%<br>afstand | De afstand vanaf de bovenrand van de *nearest positioned ancestor* of van het browservenster. |
| visibility | visible<br>hidden<br>collapse | Bepaalt of een element zichtbaar of onzichtbaar moet zijn. |
| z-index | auto<br>nummer | Bepaalt de stapelorde. |

# 4 Blok en inline-elementen

De HTML-elementen kunnen verdeeld worden in blokelementen en inline-elementen. Elk van deze elementen kan je dan met CSS opmaken. Om een modern ogende webpagina te bekomen zal naast de opmaak ook de positie van elk structuurelement moeten bepaald worden. Dit kan op verschillende manieren: statisch, drijvend, absoluut, relatief en vast.

De standaardinstelling is `position: static`. De lay-out gezien in vorige hoofdstukken gebruikte steeds impliciet deze positionering. In de volgende delen worden de alternatieven besproken.

# 5 Drijvende opmaak

Als je gebruik maakt van de **float**-eigenschap creëer je een drijvende of vlottende opmaak. Wanneer een *float*-waarde wordt meegegeven aan een element bepaalt dit niet enkel de positie van dit element, maar eveneens het gedrag van de andere elementen rond dit element. Indien je een box bijvoorbeeld links plaatst met *float: left* dan zorgt dit ervoor dat de daaropvolgende elementen zich rechts van de box zullen bevinden.

Het is mogelijk om gelijktijdig *float: left* en *float: right* te gebruiken op een pagina. De resterende HTML-code zal zich dan daar tussen wringen. Indien het niet gewenst is dat het volgend element zich rond de box plaatst kan je dit voorkomen met clear. Bij *clear: left* zal het volgend element zich onder de links drijvende box plaatsen, bij *clear: right* onder de rechts drijvende box. Indien helemaal geen opvulling gewenst is tussen de boxen gebruik je best *clear: both*.


# 6 Absolute opmaak

Het absoluut gepositioneerde element positioneert zich relatief ten opzichte van haar *nearest positioned ancestor*, haar eerste ancestor element dat niet statisch is gepositioneerd. Als er zo geen ancestor element is, zal het zich relatief ten opzichte van de `<html>`-container positioneren.

Een element met *position: absolute* gedraagt zich als een aparte laag en oefent niet langer een invloed uit op andere elementen, waardoor de volgorde van de absoluut gepositioneerde HTML-elementen helemaal geen rol meer speelt.

In het onderstaand voorbeeld is dit duidelijk. De tekst die na de twee boxen komt, houdt helemaal geen rekening met de aanwezigheid daarvan. Om dit probleem op te lossen kan men proberen alle informatie van de webpagina in een aparte laag te plaatsen.

De positie wordt bepaald door de afstand ten opzichte van rand van het browservenster, omdat er geen gepositioneerde ancestor is, dus met de *top*, *bottom*, *left* en *right* eigenschappen. Bij het wijzigen van de grootte van het browservenster blijft deze afstand ongewijzigd. Een vaste breedte kan meegegeven worden door een *width*-waarde op te geven. Indien je voor alle lagen een vaste breedte instelt zal deze niet wijzigen als het browservenster wijzigt.

Om gedeeltelijk tegemoet te komen aan wijzigingen aan het browservenster kan je ook minimum- en maximumbreedtes en -hoogten definiëren met *min-width*, *max-width*, *min-height* en *max-heigtht*.

```
#box1 {
    position: absolute;
    top: 10px;
    left: 10px;
    width: 200px;
}
#box2 {
    position: absolute;
    top: 10px;
    right: 10px;
    width: 200px;
}
#inhoud {
    position: absolute;
    top: 10px;
    right: 240px;
    left: 240px;
}
```

```
<aside id="box1">
  <p>…</p>
</aside>
<aside id="box2">
  <p>…</p>
</aside>
<article id="inhoud">
  <h1>…</h1>
  <p>…</p>
</article>
```

Als je toch het browservenster volledig wenst te benutten kan je van één of enkele lagen de breedte niet meegeven. Om deze laag toch vrij te houden ten opzichte van de andere lagen moet je de afmeting van deze lagen meerekenen in de *top*, *bottom*, *left* en *right* -waarde. Hiervoor moet je rekening houden met alle parameters, dus zowel *width*, *padding* en *margin*.

In onderstaand voorbeeld wordt de *right*- en *left*-waarde van de inhoudlaag ingesteld op 240 pixels. Dit wordt als volgt bekomen: startend van links zijn de eerste 10 pixels afkomstig van de left-waarde van box1, de volgende 2 pixels komen van de boorddikte, dan 10 pixels voor de linkerpadding, dan 200 pixels van de breedte van box1, opnieuw 10 pixels voor de rechterpadding en 2 pixels voor de rechterboord van box1. Dit geeft een totaal van 234 pixels. Aangezien de *left*-waarde van de inhoudlaag op 240 pixels ingesteld staat zijn er dus 6 pixels witruimte tussen beide lagen.


Een extra voorbeeld met een 3-kolommen krantenopmaak illustreert dit beter. Door de linker- en rechterkolom een vaste breedte mee te geven en dit niet te doen voor de middenkolom past deze kolom zich aan aan de breedte van het browservenster.


Indien voor de middenkolom wel een vaste breedte opgegeven wordt zal er een witruimte komen aan de rechterzijde. Om dit te bekomen dien je alle lagen ten opzichte van de linkerkant uit te lijnen. Voor de middenkolom tel je dus van de linkerkolom de 10 pixels linkermarge en de 250 pixels breedte samen. Hierbij tel je dan extra 20 pixels om een witruimte tussen de twee kolommen te bekomen. Voor de middenkolom geeft dit dus *left: 280px*.

Ook dient de titellaag een vaste breedte mee te krijgen door de breedtes van de 3 kolommen op te tellen, samen met de gewenste witruimtes. Om een krantenopmaak te bekomen is voor alle kolommen ook *text-align: justify* meegegeven.

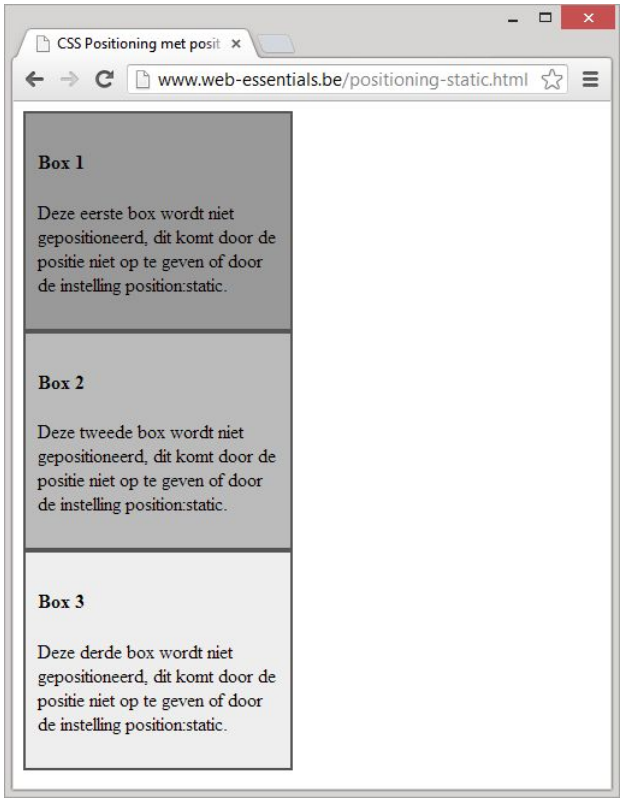
```
#top {
    position: absolute;
    top: 10px;
    left: 10px;
    width: 790px;
    height: 30px;
    padding: 5px;
    border: 1px solid black;
    text-align: center;
}
.kolom {
    position: absolute;
    text-align: justify;
    top: 60px;
    width: 250px;
}
#links {
    left: 10px;
}
#midden {
    left: 280px;
}
#rechts {
    left: 550px;
}
```

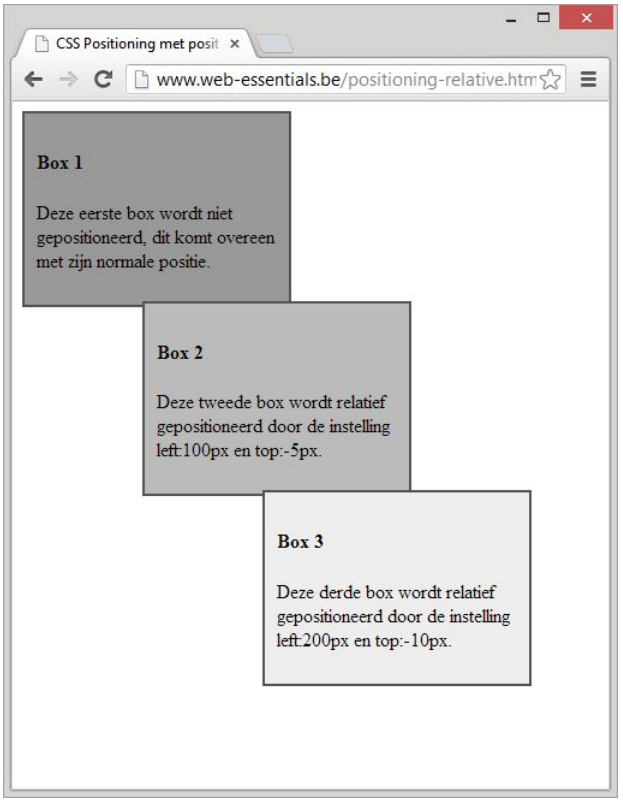

Als alternatief kan ook gebruik worden gemaakt van right voor de titel, de rechter- en de middenkolom. Door de titel en middenkolom geen width te geven, zullen deze de volledige beschikbare ruimte innemen.

# 7 Relatieve opmaak

Bij gebruik van *position: relative* verhoudt de positie van een element zich relatief ten opzichte van de normale positie die dit element zou hebben zonder positiebepaling (bij de standaardinstelling *position: static*). De normale positie wordt bepaald door de plaats van dit element in de HTML-code. Zo komen opeenvolgende blokelementen allemaal onder elkaar en inline- elementen naast elkaar.

Drie opeenvolgende div's worden normaal perfect onder elkaar weergegeven, maar door het opgeven van *position: relative* en *top*, *bottom*, *left* en *right*, kan deze positie gewijzigd worden.




Bij inline elementen geeft dit ook nieuwe mogelijkheden. Zo kan bijv. een tekstonderdeel verplaatst worden ten opzichte van zijn normale positie.

```
span {
    font-family: Arial, Helvetica, sans-serif;
    color: #999999;
    font-size: 1.2em;
}
.hoger {
    position: relative;
    bottom: 10px;
}
.lager {
    position: relative;
    bottom: -10px;
}
.links{
    position: relative;
    left: -15px;
}
.rechts {
    position: relative;
    right: -15px;
```

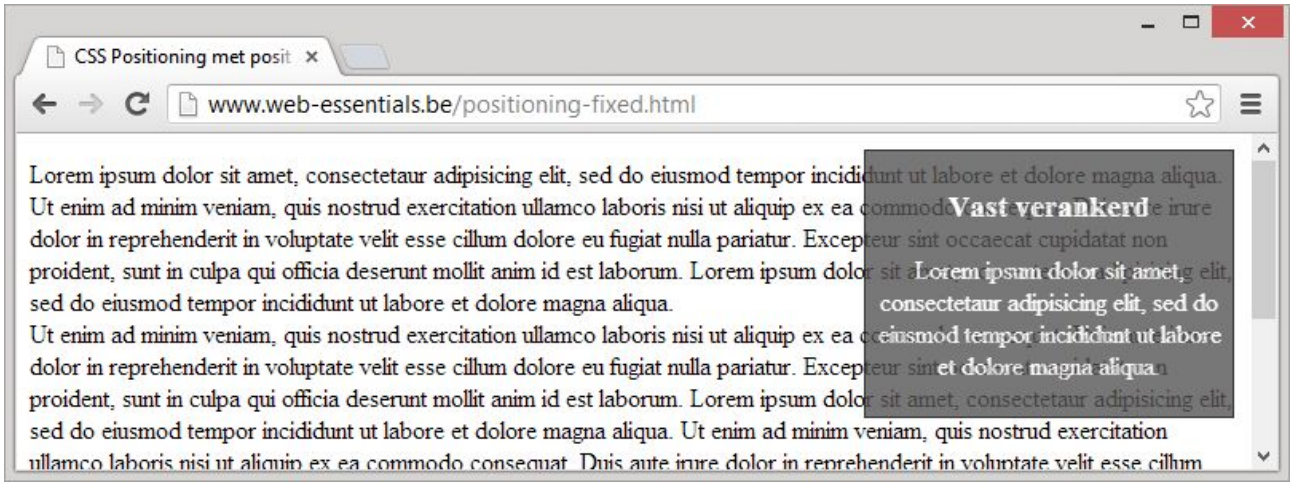
```
}
```

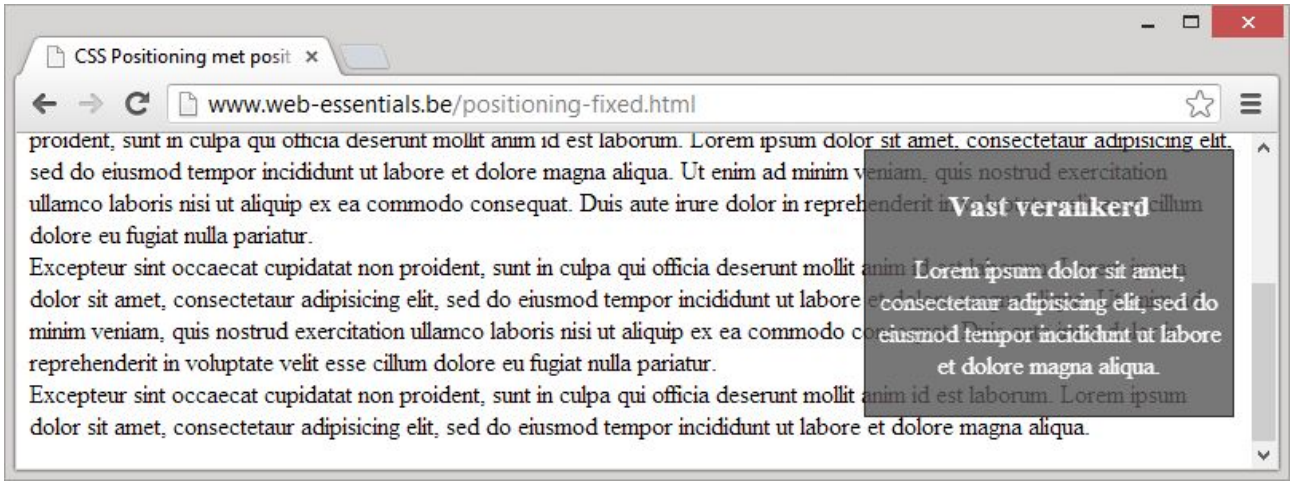
```
<p>Met de instelling position: relative kunnen inline elementen
bijvoorbeeld <span class="hoger">hoger</span> of <span
class="lager">lager</span> dan de normale tekst geplaatst worden.</p>
<p>Dit kan natuurlijk ook meer naar <span class="links">links</span> of
meer naar <span class="rechts">rechts</span> of met een <span class="links
lager">combinatie</span> van beiden.</p>
```


Merk op dat het element nog steeds ruimte in beslag neemt op haar oude positie, alsof het volgens de gewone lay-out niet verplaatst is. Dit is in tegenstelling tot `position: absolute;` waarbij het element uit de normale flow wordt gehaald en geen ruimte meer krijgt toegewezen.

# 8 Vaste positie

Met de instelling *position: fixed* worden objecten vast verankerd binnen het zichtbare veld van het browservenster. Verder werkt dit identiek aan *position: absolute*. In onderstaand voorbeeld is transparantie gebruikt om het effect te verduidelijken. Let op de positie van de schuifbalk.

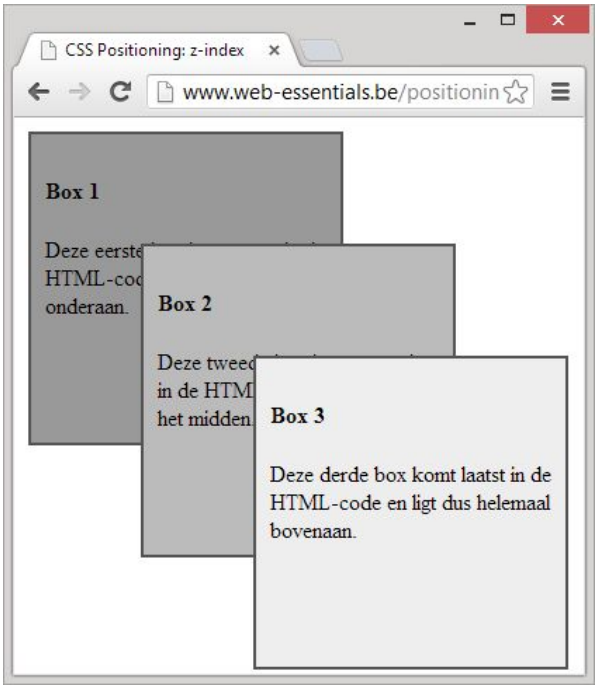


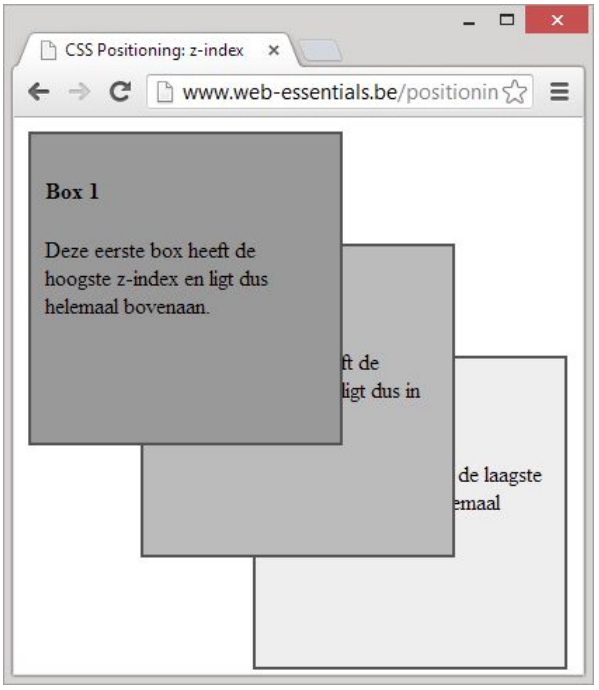



## 9 Stapelvolgorde

Zoals in bovenstaande voorbeelden met een absolute en vaste opmaak blijkt, worden de verschillende lagen op elkaar gestapeld. De laag die eerst in de HTML-code staat komt onderaan te liggen en alle volgende lagen zullen hierop gepositioneerd worden in de volgorde waarop ze in de code staan. Dit geeft niet altijd het gewenste effect en het wijzigen van de code levert een verminderde leesbaarheid op. Om manueel in te grijpen op de positie van de lagen kan de *z-index* ingesteld worden en hoeft er niet geraakt te worden aan de volgorde van de code. Objecten met een hogere z-index liggen bovenop objecten met een lager nummer. Er kunnen ook negatieve getallen gebruikt worden.

```
div {
    position: absolute;
    border: 2px solid #555555;
    padding: 10px;
    width: 200px;
    min-height: 200px;
}
#box1 {
    top: 10px;
    left: 10px;
    background-color: #999999;
    z-index: 3;
}
#box2 {
    top: 90px;
    left: 90px;
    background-color: #BBBBBB;
    z-index: 2;
}
#box3 {
    top: 170px;
    left: 170px;
    background-color: #EEEEEE;
    z-index: 1;
}
```


CSS Positioning: z-index
www.web-essentials.be/positionin
Box 1
Box 2
Box 3
Deze derde box komt laatst in de HTML-code en ligt dus helemaal bovenaan.


CSS Positioning: z-index
www.web-essentials.be/positionin
Box 1
Deze eerste box heeft de hoogste z-index en ligt dus helemaal bovenaan.