CSS boxmodel

## boxmodel

### Voorbeeld

### Te gebruiken bij:

-

### Eigenschappen:

-

### Info

#### Padding/marge

**The CSS Box Model**. Ieder doosje, ook wel *container* genoemd, bevat telkens dezelfde vier onderdelen:

De **content** zelf;

De **border** (of deze nou zichtbaar is of niet, hij is er!);

De ruimte tussen de content en de border: **padding**;

De ruimte tussen de border en het eerstvolgende element op de pagina: **margin**.

Afbeelding met tekst, schermopname, roze, Rechthoek

Automatisch gegenereerde beschrijving

Dit komt omdat de padding op een container van nature *transparant* is, totdat je de content een achtergrondkleur geeft. Laten we eens kijken wat er gebeurt als we alle elementen een blauwe achtergrondkleur geven:



Zo is het verschil ineens een stuk duidelijker! Zorg er daarom voor dat je margin altijd gebruikt voor de *afstand tot andere elementen* en padding om het element *zelf* van meer ruimte te voorzien.

#### Elementen specifieke afmetingen geven (width height )

Wellicht heb je in de wandelgangen al gehoord dat er een height en width property bestaat die we op (bijna) alle elementen kunnen toepassen. Zo kunnen we bijvoorbeeld zeggen dat onze span-elementen allemaal een hoogte van 30 pixels, een breedte van 60 pixels en wat margin en padding moeten krijgen:

span {

height: 30px;

width: 60px;

padding: 20px;

margin: 20px;

}

totaal is ons element dus **witdth + margin + padding = x pixels breed**!

box-sizing toe, zodat er niet vanaf de content wordt gerekend, maar vanaf de border

Afbeelding met Rechthoek, tekst, roze, schermopname

Automatisch gegenereerde beschrijving

Borders geven ons de mogelijkheid om het uiterlijk van onze elementen naar een hoger niveau te tillen. Wanneer je een rand wil toepassen, doe je dat altijd met drie onderdelen:

1. De dikte, met border-width;
2. De kleur, met border-color;

#### typen borders

De stijl van de border, met border-style.

Omdat je deze properties allemaal nodig hebt wanneer je een border wilt toevoegen, mag je hier ook de *shorthand* voor gebruiken, in plaats van drie losse declaraties:

div {

border: 2px solid purple;

}



Afbeelding met tekst, schermopname, Lettertype, nummer

Automatisch gegenereerde beschrijving

#### Ronde vormen

Borders geven ons ook de mogelijkheid om ronde vormen te creëren, ook als de rand zelf onzichtbaar is. In onderstaand voorbeeld geven we een aantal div-elementen afgeronde hoeken

border-radius-propery. Uiteraard is het ook mogelijk om hier nog een rand aan toe te voegen.

Afbeelding met logo, Graphics, ontwerp

Automatisch gegenereerde beschrijving

div {

width: 50px;

height: 50px;

background-color: purple;

}

.square {

border-radius: 5px;

}

.shape {

border-bottom-left-radius: 20px;

border-bottom-right-radius: 5px;

border-top-left-radius: 5px;

border-top-right-radius: 20px;

/\* Deze vier regels kunnen we ook vervangen door: \*/

/\*border-radius: 5px 20px 5px 20px; \*/

}

.circle {

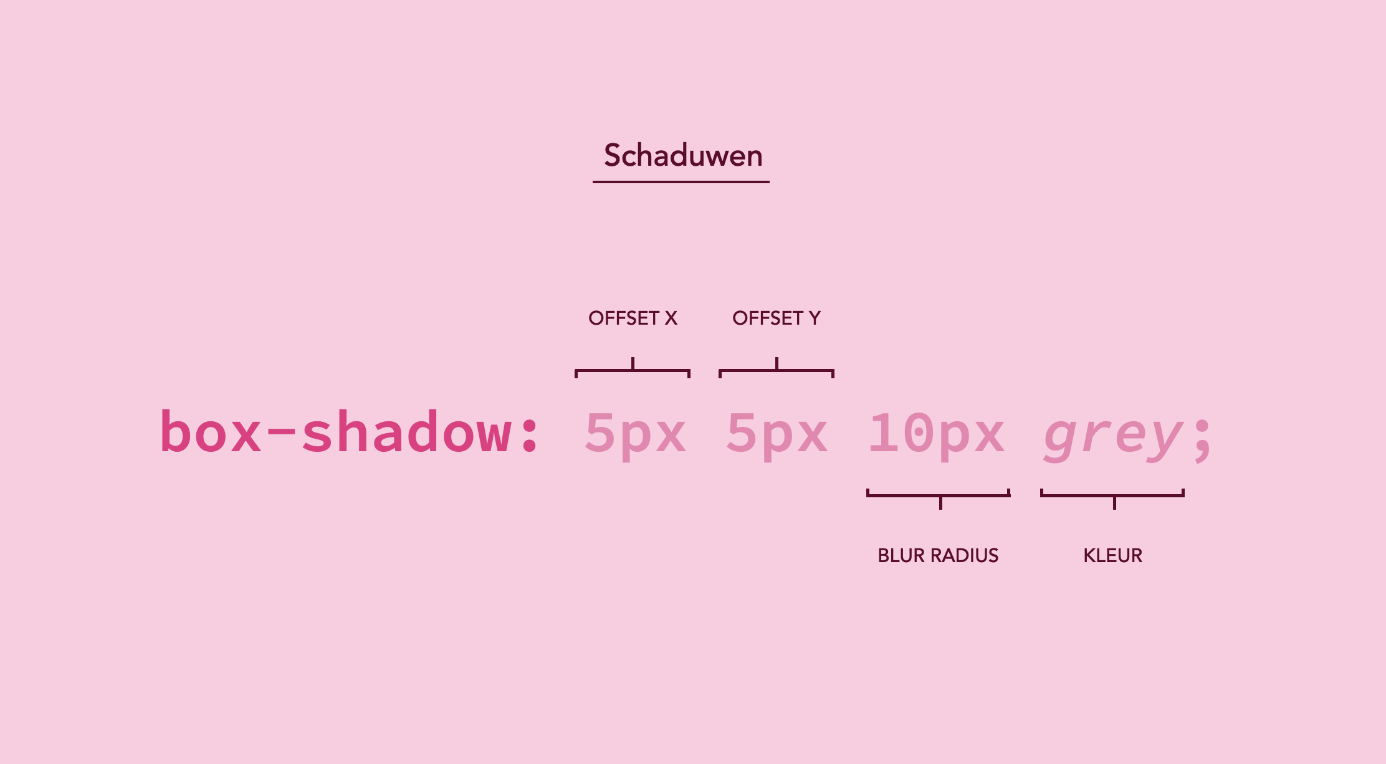
border-radius: 50%;

}

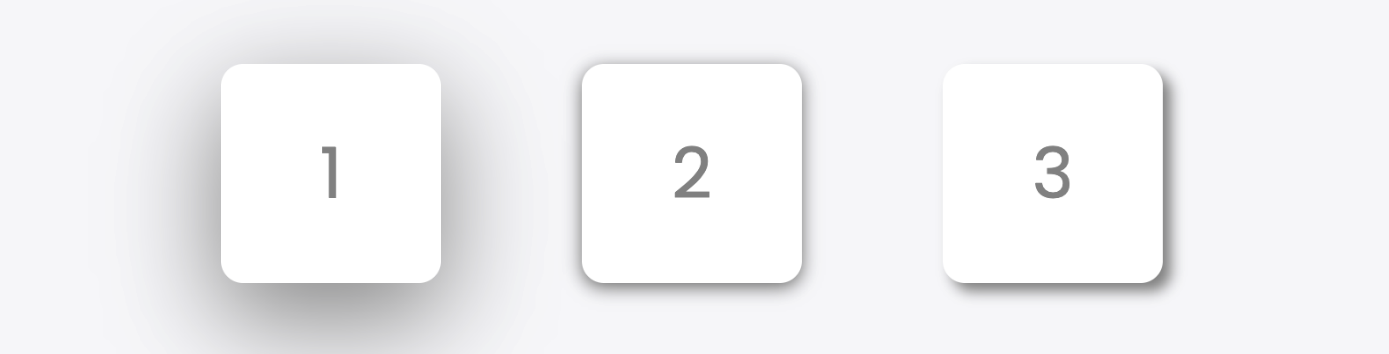
Wanneer we de border-radius op minimaal de helft van de breedte en/of hoogte declareren, wordt ons vierkant vanzelf een cirkel. Dit kun je dus ook toepassen op afbeeldingen!

#### Schaduwen

de schaduw zich horizontaal verplaatst (offset x), verticaal verplaatst (offset y), hoe hard of zacht de shadow overvloeit (blur-radius) en welke kleur deze moet hebben.



"hard" of overdreven:



Wanneer je een meer subtielere schaduw wil toepassen, zul je daarom gebruik moeten maken van kleuren in de RGBA-schrijfwijze.

Afbeelding met elektronica, stekker

Automatisch gegenereerde beschrijving

.normal {

box-shadow: 0 8px 24px rgba(0, 0, 0, 0.10);

}

.shape {

box-shadow: 0 1px 4px rgba(0, 0, 0, 0.16);

}

.circle {

box-shadow: 2px 2px 3px rgba(0, 0, 0, 0.15);

}

#### De position-property

Oké, tijd om écht te gaan knutselen met positionering. CSS heeft ons de optie om één van de volgende opties op elementen toe te passen:

* Relatieve positionering;
* Absolute positionering;
* Fixed positionering;
* Sticky positionering.

Afbeelding met tekst, brief, Lettertype, schermopname

Automatisch gegenereerde beschrijving

De drie paragrafen staan gebundeld in één omwikkelende <section>-tag. Deze heeft een lichtroze achtergrondkleur ter illustratie.

##### Relatieve positionering

Relatieve positionering verplaatst een element in relatie tot waar het had gestaan in de normale pagina-flow. Hieraan geven we dan een **offset** mee: de daadwerkelijke hoeveelheid verplaatsing van het element. Zo zou je het bijvoorbeeld 40px vanaf de bovenkant naar beneden kunnen verplaatsen en 80px vanaf de linkerkant naar rechts:

p.moved {

position: relative;

top: 40px;

left: 80px;

}

De andere elementen houden nog wel rekening met de aanwezigheid van dit element, alsof het nog op zijn oude plek zou staan:

##### Absolute positionering

##### Afbeelding met tekst, Lettertype, brief, schermopname Automatisch gegenereerde beschrijving

Absolute positionering verplaatst een element ergens ten opzichte van een ander element. Dit is **altijd** het meest nabije parent-element met position: relative. Als geen enkel element op jouw pagina deze position: relative toegewezen heeft gekregen, valt hij standaard terug op het buitenste element: de <body>-tag.

Wanneer we onze middelste paragraaf domweg absoluut positioneren zonder een ander element van position: relative te voorzien, gebeurt er het volgende:

p.moved {

position: absolute;

top: 40px;

left: 80px;

}

##### Fixed positionering;

Afbeelding met tekst, Lettertype, schermopname, brief

Automatisch gegenereerde beschrijving

Zoals je ziet wordt de middelste paragraaf nu gepositioneerd ten opzichte van het browser-window: het body-element. Ook is duidelijk te zien dat absolute positionering het element uit de normale flow van de pagina haalt. Andere elementen houden geen rekening meer met de aanwezigheid van dit element.

Wanneer we onze omwikkelende parent (het section-element) een position: relative; geven, komt de middelste paragraaf netjes 40px onder de bovenste rand en 80px van de linker rand van het lichtroze vlak af te staan:

p.moved {

position: absolute;

top: 40px;

left: 80px;

}

p.first {

position: relative;

}

Z-index

De **z-index** speelt een belangrijke rol in samenwerking met de laatste twee positionerings-properties. De z-index noemen we ook wel de stapel-index, omdat het refereert naar de volgorde waarop elementen op elkaar gestapeld zijn. Hoe hoger het cijfer, hoe hoger het element op de stapel ligt. Als een element onder een ander element verdwijnt, kun je het naar voren halen door hem een hogere z-index te geven. Standaard hebben alle elementen een z-index van 0.

Fixed positionering

Fixed positionering haalt het element ook uit de normale flow van de pagina, waardoor andere elementen geen rekening meer houden met de aanwezigheid van dit element. De offset properties plaatsen het element altijd ten opzichte van de body. Dus als de gebruiker scrolt, blijft het element op precies dezelfde plek staan.

Wanneer we onze titel in een <header>-tag plaatsen, kunnen we een achtergrondkleur en width- en height-properties toepassen, zodat het duidelijker zichtbaar is dat de titel bovenaan het scherm gepositioneerd blijft.

header {

position: fixed;

top: 0;

left: 0;

height: 80px;

width: 100%;

background-color: #C84E80;

color: white;

}

Afbeelding met tekst, schermopname, Lettertype, ontwerp

Automatisch gegenereerde beschrijving

##### Sticky positionering

Sticky positionering lijkt in veel opzichten identiek te zijn met fixed positionering, maar er zijn een twee kleine verschillen:

Fixed zet het element vast ten opzichte van het browser-scherm.

Sticky scrolt mee alsof het een normaal element is, tot het de offset bereikt die jij hebt gedeclareerd. Vanaf dat moment doet het alsof het fixed is.

Er zit een kleine catch aan: een sticky element **verlaat zijn containing-parent nooit**. Op het moment dat het container-element van de pagina "afgescrold" wordt, verlaat het sticky element de pagina ook. Het volgende voorbeeld illustreert dit:

div.sticky {

position: sticky;

top: 0;

}

Afbeelding met schermopname, Rechthoek, visitekaartje, plein

Automatisch gegenereerde beschrijving

Let op: bij het declareren van offset properties kun je slechts één horizontale en één verticale offset mee te geven. Dus bijvoorbeeld top en right of bottom en left.

#### Aantekening

HTML-voorbeeld:

html

<div class="container">

<div class="box relative">Relative Positioning</div>

<div class="box absolute">Absolute Positioning</div>

<div class="box fixed">Fixed Positioning</div>

<div class="box sticky">Sticky Positioning</div>

</div>

CSS-voorbeeld:

css

Copy code

.container {

height: 300px;

position: relative;

}

.box {

width: 200px;

height: 100px;

background-color: #f2f2f2;

border: 1px solid #ddd;

padding: 10px;

margin-bottom: 10px;

}

.relative {

position: relative;

top: 20px;

left: 20px;

}

.absolute {

position: absolute;

top: 50px;

right: 20px;

}

.fixed {

position: fixed;

top: 100px;

left: 20px;

}

.sticky {

position: sticky;

top: 150px;

right: 20px;

}

Uitleg:

In het HTML-voorbeeld hebben we een container-div met daarin vier div-elementen met de klasse "box", elk met een andere positioneringsoptie.

De CSS-stijlen worden toegepast op de box-elementen om de positionering te beheren.

Relative positionering (relative): Hiermee kan een element worden verplaatst ten opzichte van zijn normale positie. De top- en left-eigenschappen worden gebruikt om de positie aan te passen. In het voorbeeld wordt de eerste box 20 pixels naar rechts en 20 pixels naar beneden verplaatst.

Absolute positionering (absolute): Hiermee kan een element absoluut worden gepositioneerd ten opzichte van zijn eerste voorouder met een positionele eigenschap (niet statisch). De top- en right-eigenschappen worden gebruikt om de positie aan te passen. In het voorbeeld wordt de tweede box 50 pixels naar beneden en 20 pixels naar rechts verplaatst ten opzichte van de container-div.

Fixed positionering (fixed): Hiermee kan een element absoluut worden gepositioneerd ten opzichte van het viewport (de zichtbare ruimte van de browser). De top- en left-eigenschappen worden gebruikt om de positie aan te passen. In het voorbeeld blijft de derde box altijd op dezelfde positie, zelfs als er wordt gescrolld.

Sticky positionering (sticky): Hiermee kan een element zich gedragen als een relatief gepositioneerd element totdat het een bepaalde drempel bereikt, waarna het zich gedraagt als een vast gepositioneerd element. De top- en right-eigenschappen worden gebruikt om de positie aan te passen. In het voorbeeld blijft de vierde box aan de bovenkant van het scherm plakken zodra de scrollpositie de drempel van 150 pixels bereikt.