# Boxmodel en flexbox

### Voorbeeld

### Te gebruiken bij:

-

### Eigenschappen:

-

### Info

Afbeelding met tekst, schermopname, ontwerp

Automatisch gegenereerde beschrijving

#### boxmodel

Het boxmodel is een concept in CSS (Cascading Style Sheets) dat wordt gebruikt om de lay-out en de positionering van elementen op een webpagina te beheren. Het boxmodel verdeelt elk HTML-element in een rechthoekig "box"-gebied dat bestaat uit randen, marges, padding en de eigenlijke content van het element. Hieronder zal ik het boxmodel uitleggen en enkele voorbeelden geven in zowel code als tekst.

Het boxmodel bestaat uit vier delen:

Content: Dit is het daadwerkelijke zichtbare gebied van het element waar de inhoud, zoals tekst of afbeeldingen, wordt weergegeven. De grootte van de content wordt bepaald door de breedte en hoogte die aan het element zijn toegewezen.

Padding: De padding is het gebied rond de content binnen de grenzen van het element. Het kan worden gebruikt om extra ruimte tussen de content en de randen van het element te creëren. Het beïnvloedt de totale grootte van het element niet.

Border: De rand is een lijn die het gebied rond de padding markeert. Het kan worden gestyled en van verschillende diktes en stijlen worden voorzien. Het volgt de padding en beïnvloedt de grootte van het element.

Margin: De marge is het gebied buiten de grenzen van het element. Het wordt gebruikt om ruimte te creëren tussen het element en andere elementen in de lay-out. Marges hebben geen invloed op de grootte van het element zelf.

Hier is een voorbeeld van hoe het boxmodel kan worden toegepast in HTML en CSS:

html

<!DOCTYPE html>

<html>

<head>

<style>

.box {

width: 200px;

height: 100px;

padding: 20px;

border: 2px solid black;

margin: 10px;

}

</style>

</head>

<body>

<div class="box">

Dit is de inhoud van de box.

</div>

</body>

</html>

In dit voorbeeld hebben we een div-element met de class "box". We hebben een breedte en hoogte van 200 pixels en 100 pixels toegewezen aan dit element. Daarnaast hebben we 20 pixels padding, een 2 pixels brede solide zwarte rand en een marge van 10 pixels.

De totale grootte van het element, inclusief de inhoud, padding, border en marge, wordt bepaald door de som van deze waarden. In dit geval is de totale breedte van de box: 200px (content) + 2 \* 20px (padding links en rechts) + 2 \* 2px (border links en rechts) + 2 \* 10px (margin links en rechts) = 264px. De totale hoogte van de box is: 100px (content) + 2 \* 20px (padding boven en onder) + 2 \* 2px (border boven en onder) + 2 \* 10px (margin boven en onder) = 164px.

1. Width en height: Dit zijn eigenschappen waarmee je de breedte en hoogte van een element kunt instellen. Je kunt specifieke waarden opgeven, zoals pixels (px) of percentages (%), om de grootte van het element te bepalen.
2. Margin: De marge is het gebied buiten de randen van het element. Het kan worden gebruikt om ruimte te creëren tussen elementen. De marges kunnen individueel worden ingesteld voor elke zijde van het element (bijv. margin-top, margin-right, margin-bottom, margin-left) of in één enkele eigenschap (bijv. margin: 10px;).
3. Padding: De padding is het gebied tussen de randen van het element en de content binnenin. Het wordt gebruikt om ruimte te creëren tussen de content en de randen. Net als de marges kan de padding individueel worden ingesteld voor elke zijde (bijv. padding-top, padding-right, padding-bottom, padding-left) of in één enkele eigenschap (bijv. padding: 10px;).
4. Border: De rand is een lijn rond het element dat de grens markeert tussen de padding en de marge. Het kan worden gestyled met eigenschappen zoals border-width, border-style en border-color om de dikte, stijl en kleur van de rand aan te passen.
5. Box-sizing: Dit is een eigenschap waarmee je kunt bepalen hoe de totale grootte van het element wordt berekend. De standaardwaarde is **content-box**, wat betekent dat de breedte en hoogte alleen de content bevatten. Met de waarde **border-box** wordt de breedte en hoogte inclusief padding en randen berekend, zodat de content binnen de opgegeven afmetingen blijft.
6. Overflow: Deze eigenschap bepaalt wat er gebeurt als de inhoud van een element niet past binnen de opgegeven breedte en hoogte. De waarde **visible** geeft aan dat de inhoud buiten het element kan uitsteken, **hidden** zorgt ervoor dat de inhoud wordt verborgen, **scroll** toont een schuifbalk en **auto** toont een schuifbalk indien nodig.

#### flexbox

Flexbox is een CSS-layoutmodule waarmee je flexibele en responsieve lay-outs kunt creëren. Het biedt een efficiënte manier om elementen te positioneren, te ordenen en de beschikbare ruimte te verdelen binnen een container. Hieronder zal ik het flexbox-model uitleggen en enkele voorbeelden geven in zowel code als tekst.

Flexbox maakt gebruik van een flex-container en flex-items. De flex-container is de ouder-container die de flex-items bevat. Het heeft eigenschappen waarmee je de lay-out en het gedrag van de flex-items kunt beheren. De flex-items zijn de kind-elementen van de flex-container en worden geplaatst en aangepast binnen de beschikbare ruimte.

Hier zijn enkele belangrijke eigenschappen die worden gebruikt bij het werken met flexbox:

display: flex;: Dit wordt gebruikt om een element als flex-container in te stellen. Alle directe kinderen van dit element worden automatisch flex-items.

flex-direction: row/column;: Dit bepaalt de richting waarin de flex-items worden geplaatst. Bij 'row' worden de items horizontaal geplaatst (van links naar rechts) en bij 'column' verticaal (van boven naar beneden).

justify-content: flex-start/center/flex-end/space-between/space-around;: Dit bepaalt hoe de flex-items langs de hoofdas (de richting van flex-direction) worden uitgelijnd. Bijvoorbeeld 'flex-start' plaatst de items aan het begin van de as, 'center' in het midden en 'flex-end' aan het einde.

align-items: flex-start/center/flex-end/stretch;: Dit bepaalt hoe de flex-items langs de dwarsas (de tegenovergestelde richting van flex-direction) worden uitgelijnd. Bijvoorbeeld 'flex-start' plaatst de items aan de bovenkant, 'center' in het midden en 'flex-end' aan de onderkant.

flex-grow, flex-shrink, flex-basis: Deze eigenschappen bepalen hoe de flex-items de beschikbare ruimte binnen de flex-container verdelen. flex-grow bepaalt de verhouding waarin de items groeien, flex-shrink bepaalt de verhouding waarin de items krimpen en flex-basis bepaalt de basisgrootte van de items voordat de resterende ruimte wordt verdeeld.

Hier is een voorbeeld van hoe flexbox kan worden toegepast in HTML en CSS:

html

<!DOCTYPE html>

<html>

<head>

<style>

.container {

display: flex;

flex-direction: row;

justify-content: space-between;

align-items: center;

}

.item {

flex: 1;

height: 100px;

margin: 10px;

background-color: lightblue;

}

</style>

</head>

<body>

<div class="container">

<div class="item">Item 1</div>

<div class="item">Item 2</div>

<div class="item">Item 3</div>

</div>

</body>

</html>

In dit voorbeeld hebben we een .container-element dat is ingesteld als een flex-container met display: flex;. De flex

Flex-container: Dit is het element dat wordt ingesteld met **display: flex;**. Het fungeert als een container voor flex-items en bepaalt de lay-out van de flex-items binnen zijn ruimte.

Flex-item: Dit zijn de kind-elementen van de flex-container. Ze worden automatisch flex-items zodra hun ouder-element is ingesteld als een flex-container. Flex-items worden gepositioneerd en aangepast binnen de beschikbare ruimte van de flex-container.

Flex-direction: Deze eigenschap bepaalt de richting waarin de flex-items worden geplaatst. Het kan worden ingesteld op **row** voor een horizontale lay-out (van links naar rechts) of **column** voor een verticale lay-out (van boven naar beneden).

Justify-content: Deze eigenschap bepaalt hoe de flex-items worden uitgelijnd langs de hoofdas (de richting bepaald door flex-direction). Het beïnvloedt de horizontale uitlijning van de items. Enkele waarden zijn:

**flex-start**: De items worden uitgelijnd aan het begin van de as.

**center**: De items worden gecentreerd op de as.

**flex-end**: De items worden uitgelijnd aan het einde van de as.

**space-between**: De items worden gelijkmatig verdeeld met ruimte ertussen.

**space-around**: De items worden gelijkmatig verdeeld met ruimte rondom.

Align-items: Deze eigenschap bepaalt hoe de flex-items worden uitgelijnd langs de dwarsas (de tegenovergestelde richting van flex-direction). Het beïnvloedt de verticale uitlijning van de items. Enkele waarden zijn:

**flex-start**: De items worden uitgelijnd aan de bovenkant van de dwarsas.

**center**: De items worden gecentreerd op de dwarsas.

**flex-end**: De items worden uitgelijnd aan de onderkant van de dwarsas.

**stretch**: De items worden uitgerekt om de volledige hoogte van de dwarsas te vullen.

Flex-grow: Deze eigenschap bepaalt hoe de beschikbare ruimte binnen de flex-container wordt verdeeld onder de flex-items als er extra ruimte is. Het wordt gebruikt om aan te geven hoeveel een flex-item moet groeien in verhouding tot andere items.

Flex-shrink: Deze eigenschap bepaalt hoe de flex-items krimpen wanneer er niet genoeg ruimte is binnen de flex-container. Het wordt gebruikt om aan te geven hoeveel een flex-item moet krimpen in verhouding tot andere items.

Flex-basis: Deze eigenschap bepaalt de basisgrootte van een flex-item voordat de resterende ruimte wordt verdeeld. Het kan worden ingesteld als een absolute lengte of een percentage van de beschikbare ruimte.

Aspect-ratio: de aspect-ratio eigenschap wordt gebruikt om de beeldverhouding van een element te definiëren. Het stelt je in staat om de breedte en hoogte van een element te verhouden tot een bepaalde verhouding, onafhankelijk van de daadwerkelijke afmetingen van het element. Het is vooral handig bij het werken met responsive ontwerpen, waarin je de verhoudingen van elementen wilt behouden, zelfs wanneer de schermgrootte verandert.

Hier is een voorbeeld van hoe je de aspect-ratio eigenschap kunt gebruiken:

css

.container {

aspect-ratio: 16/9; /\* Stel de beeldverhouding in op 16:9 \*/

width: 100%; /\* Zorg ervoor dat de container de volledige breedte van de ouder inneemt \*/

}

In dit voorbeeld wordt de aspect-ratio eigenschap toegepast op een container element met de classnaam .container. De beeldverhouding is ingesteld op 16:9, wat een veelvoorkomende beeldverhouding is voor video's. Hierdoor behoudt het element altijd dezelfde verhouding, ongeacht de daadwerkelijke afmetingen.

Merk op dat de aspect-ratio eigenschap momenteel nog in de CSS-specificatie staat, maar het wordt nog niet breed ondersteund door alle browsers. Het is raadzaam om de browsercompatibiliteit te controleren voordat je deze eigenschap gebruikt in een productieomgeving.