> *I'll be more enthusiastic about encouraging thinking outside the box when there's evidence of any thinking going on inside it.*
> — Terry Pratchett

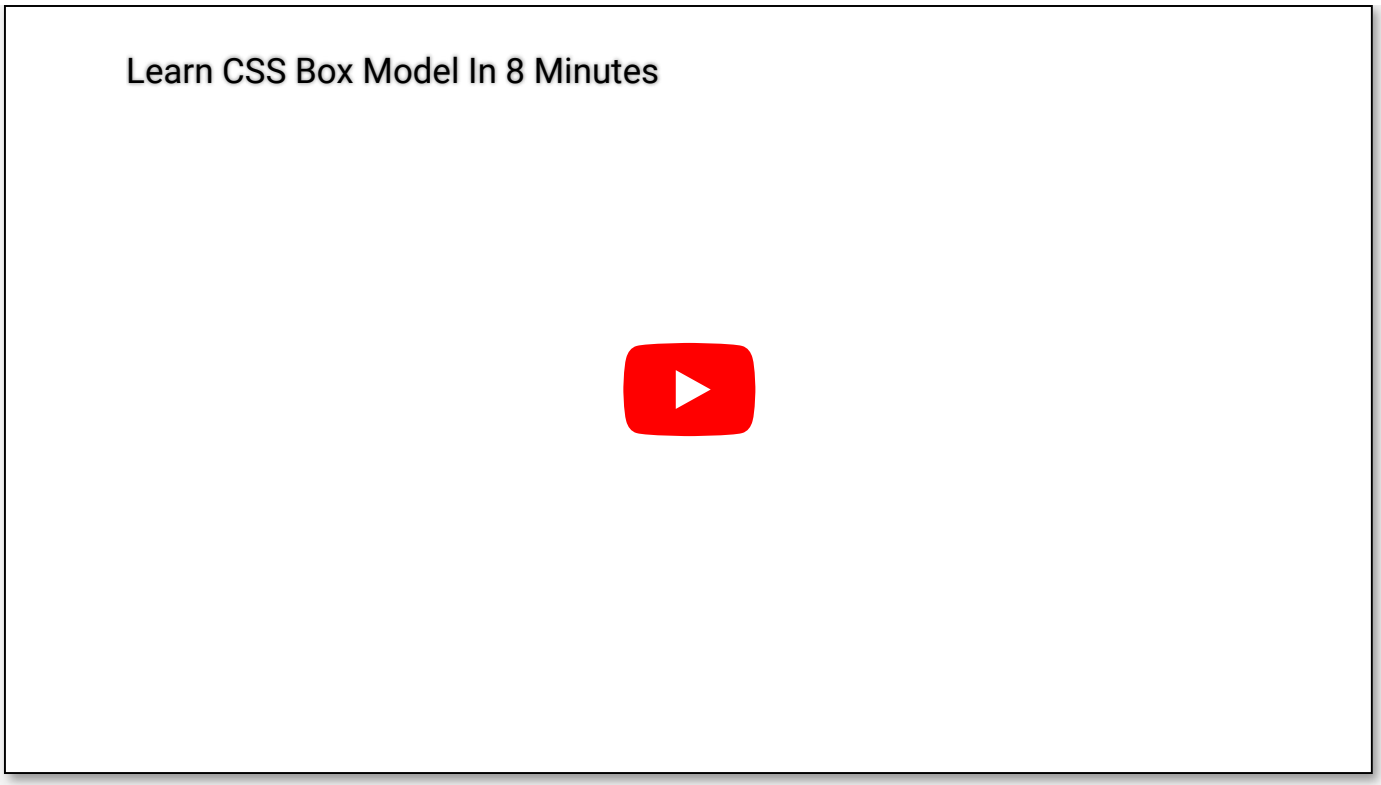
CSS box model

1 Box model in 8 minuten

In onderstaande video geeft Kyle een kort overzicht van het doosmodel (*box model*) van CSS. Aangezien alles in CSS een doos is, is het belangrijk dat je dit goed snapt. In de video gaat het over `width`, `height`, `padding`, `border`, `margin`, collapsing margins (opgelet: enkel verticaal tussen twee dozen) en de `box-sizing` eigenschap.

Bij deze video hoort een klein 'cheat sheet', te downloaden in lichte of donkere versie (PDF, 1,6 MB).

Na de video kom ik nog wat meer in detail op een aantal punten terug.



2 Eenheden

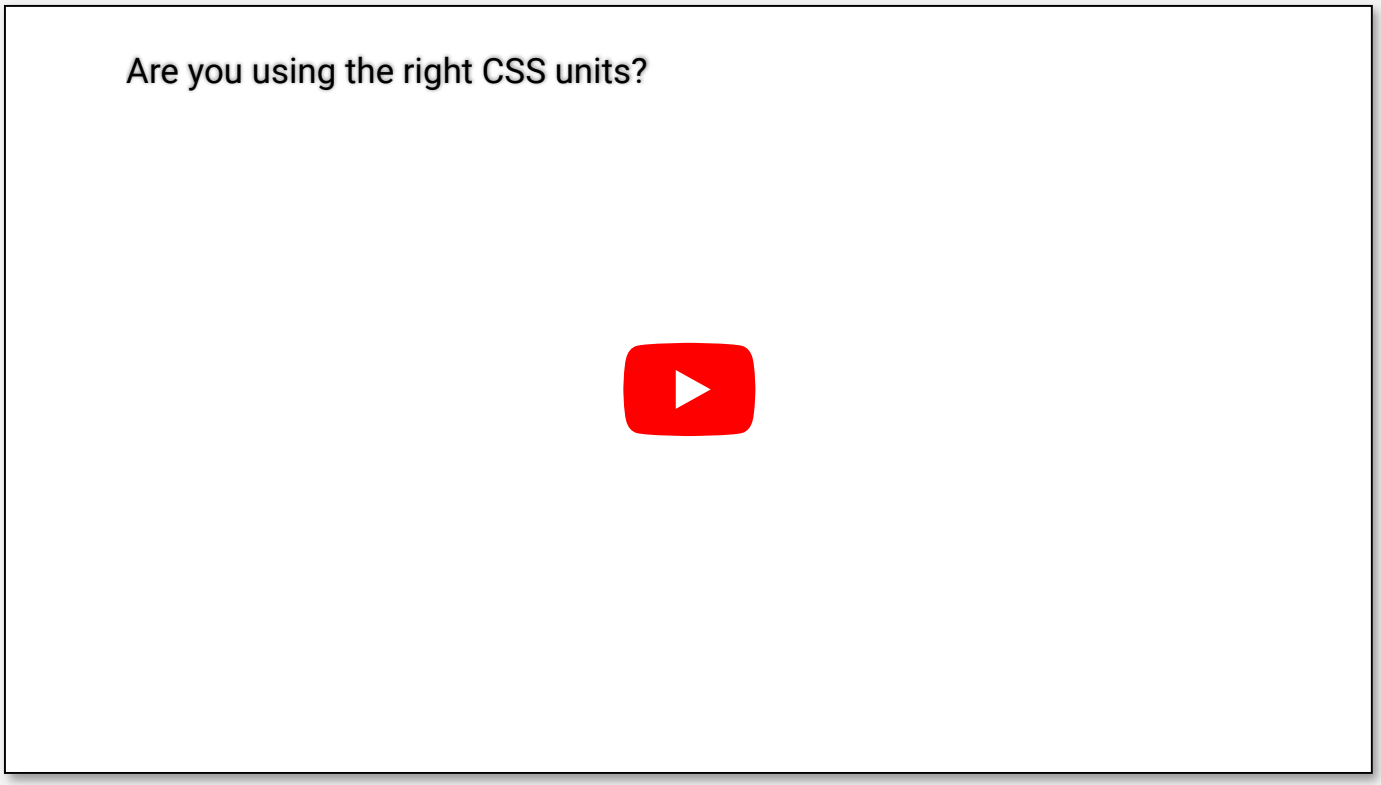
We gebruikten ondertussen al geregeld de `px` eenheid. Dat is een gemakkelijke eenheid, met echter één groot nadeel: in de wereld van RWD is heel veel vloeibaar en verwacht je dat dingen zich aanpassen aan de beschikbare grootte (van het scherm, de container enz;). De eenheid `px` maakt alles echter vast. Eenheden zoals `%` of `(r)em` zijn veel meer geschikt voor deze taak.

> ⚠ Hiermee zeg ik niet dat je nooit `px` mag gebruiken! Maar het heeft ook te maken met het laten schieten van absolute controle en rekening houden met toegankelijkheid. Zo kan je bvb de grootte van het lettertype van een stukje tekst definiëren als `font-size: 18px;`. Daar is op zich niets mis mee. Als je echter de definitie herschrijft als `font-size:1.1rem` geef je eigenlijk een stukje controle terug aan de gebruiker: “dit stukje tekst moet getoond worden in een lettergrootte die 1,1 keer zo groot is als de standaardlettergrootte die de gebruiker ingesteld heeft in de voorkeuren van haar browser”. Deze standaardgrootte is gewoonlijk 16px, dus 1,1*16px ligt heel dicht bij 18px. Maar mogelijk wordt je site bekeken door iemand die niet zo goed ziet en in de browservoorkeuren het standaarlettertype een stuk groter gezet heeft.

- `em` Lettergrootte van het element
- `ex` hoogte van de kleine letters

`cap` hoogte van de hoofdletters
`ch` Gemiddelde breedte van een letter (‘0’)
`rem` Lettergrootte van het root element
`vw` 1% van de viewport's breedte
`vh` 1% van de viewport's hoogte
`vmin` 1% van de kleinste afmeting (b / h) van de viewport
`vmax` 1% van de grootste afmeting van de viewport
`%` percentage, altijd t.o.v. een andere grootheid

Kevin Powell heeft naar goede gewoonte een overzichtelijk filmpje met heel veel goed advies over welke eenheid je nu juist gebruikt.



3 em vs rem?

Een vraag die veel studenten stellen: wat is nu juist het verschil tussen `em` en `rem`? Kevin geeft eerst in een klein filmpje een use case waar `em` te verkiezen kan zijn boven `rem`. Onder het filmpje volgt een kleine oefening die laat zien hoe lettergroottes in `em` en `rem` kunnen verschillen.

em vs rem - css units | #shorts

Gegeven volgend stukje code. Voor het gemak hebben we de CSS in het `style` element van `head` gezet (je weet dat dat niet de voorkeursmethode is, hee?). Zo hebben we maar één bestand nodig om deze oefening te maken.

```
<!DOCTYPE html>
<html lang="nl">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Verschil em / rem</title>
    <style>
        section {
            font-size: 2rem; /* wat met em? */
        }

        p {
            font-size: 1.5rem; /* verander in em */
        }
    </style>
</head>
<body>
    <section>
        <p>Lorem ipsum dolor sit.</p>
    </section>
</body>
</html>
```

Veronderstel dat het standaardlettertype van de browser 16px groot is. Beantwoord nu volgende vragen:

1. Hoe groot (px) zal de `font-size` van de tekst in de `p` zijn?
2. Vervang nu twee keer de eenheid `rem` door `em`. Hoe groot is de tekst nu?

Toon / verberg oplossing

4 Breedte en hoogte

Bovenop wat er hierboven in het filmpje over het box model verteld is over `width` en `height` nog deze twee opmerkingen:

1. De berekeningen worden een heel stuk eenvoudiger als je volgende CSS-regel boven aan je stijlblad schrijft: `* {box-sizing: border-box}`
2. Zoals Kevin Powell ook al aangeeft: moet je echt de hoogte van een doos bepalen? Waarom niet automatisch: hoe meer inhoud er in die doos komt, des te hoger zal ze worden. Als je zelf de hoogte bepaalt, loop je het risico op *overflow*. Als je dan toch een hoogte moet instellen, is het vaak veiliger om `min-height` te gebruiken.

5 Padding / margin

Een veelgebruikte toepassing van `padding` is zorgen dat letters nooit tot tegen de rand van een doos komen. Je kan dit natuurlijk ook bereiken met `margin`, maar vaak is het gemakkelijker om in je element wat witruimte te voorzien met `padding`

6 Border

Randen rond doosjes zijn vaak niet zo mooi. Als je met uitlijning of een achtergrondkleur werkt, zien mensen automatisch randen zonder dat je die expliciet tekent.

Tijdens het ontwerpen is het vaak wel een goed idee om je randen zichtbaar te maken. Liefst zelfs in opzichtige kleuren (voor deze drie dozen `h2` en twee keer `p` zijn volgende kleuren gebruikt: `#0f0`, `#f00` en `#00f`). Zo zie je tot waar de

verschillende elementen gaan. Een handige CSS-afkorting in Emmet is hier `bd` gevolgd door tab-toets of enter. Deze afkorting geeft als resultaat: `border: 1px solid #000;`. Het volstaat dan om één of meer nullen in deze kleurcode voor zwart te veranderen in 'F' om een felrode, -groene, ... kleur te bekomen.

7 Box drop shadows

Met de `box-shadow` CSS-eigenschap kan je een zogenaamde 'drop shadow' bij een doos maken, zodat je de indruk krijgt dat dat element lichtjes boven het blad zweeft. CSS-tricks heeft een goede uitleg over `box-shadow`.

De eigenschap `box-shadow` (ook `text-shadow` trouwens, samen met nog wel wat dingen ... ik kijk naar jou `border-radius`) is één van die eigenschappen waar beginnende developers / ontwerpers zich nogal eens aan vergrijpen (ik pleit mee schuldig, het blokje waar je nu naar kijkt heeft een schaduw ...). Als je dit gebruikt, doe het dan met mate. Hou je in! Niets zo lelijk als een opzichtige drop shadow.

↑ Naar top