# @media query’s

### Voorbeeld

@media screen (min-width: 576px)

*"Start with the small screen first, then expand until it looks like shit. Time for a breakpoint!"*

[Stephen Hay’s](https://bradfrost.com/blog/post/bdconf-stephen-hay-presents-responsive-design-workflow/)

### Te gebruiken bij:

-

### Eigenschappen:

-

### Info

Media-type

@media

Het media-type geeft aan voor welke generieke apparaat types de regels van kracht worden. Als we geen media-type zouden specificeren, selecteert de @media-regel standaard alle types. Het type komt direct achter de @media-regel te staan. Hierin onderscheiden we drie categorieën:

* print — voor documenten die worden weergegeven op een scherm in print-preview modus;
* screen — voor computerschermen, tablets en smartphones;
* speech — voor screen readers die de pagina hardop voorlezen;

In ons geval zullen we bijna altijd gebruik maken van de optie voor schermen. Dit schrijven we zo:

@media screen and (min-width: 576px) {

p {

font-size: 12px;

}

}

#### Breakpoints

De maximale of minimale hoeveelheid pixels wordt altijd aangegeven in **breakpoints**. In bovenstaand voorbeeld was het breakpoint 600 pixels. Tegenwoordig hebben we zo veel soorten schermgroottes voor smartphones, tablets en laptops dat het bijna niet meer bij te houden is. Om te voorkomen dat we voor elke paar pixels andere stijlregels gaan toepassen, is het handig een aantal standaarden te gebruiken. Let wel: deze ratio's staan continue ter discussie, dus het is slechts een leidraad.

* Telefoons in portrait: tot 576px;
* Telefoons in landscape: van 576px tot 768px;
* Tablets: vanaf 768px;
* Kleine computerschermen: vanaf 992px;
* Standaard computerschermen: vanaf 1200px;
* Grote computerschermen: vanaf 1600px;

Normaliter schrijf je media queries *of* allemaal in maximale waardes, *of* allemaal in minimale waardes. Dus dit:

p {

font-size: 12px;

}

@media screen and (min-width: 768px) {

p {

font-size: 14px;

}

}

@media screen and (min-width: 992px) {

p {

font-size: 16px;

}

}

werk je in de praktijk altijd met minimale waardes. Dis van klein naar groot.

/\* Alles wat men buiten de media query specificeert, geldt dan voor mobiele apparaten \*/

p {

font-size: 12px;

}

/\* Minimaal 768px betekend regels die gelden voor tablets \*/

@media screen and (min-width: 768px) {

p {

font-size: 14px;

}

}

/\* Minimaal 992px betekend regels die gelden voor kleine computerschermen \*/

@media screen and (min-width: 992px) {

p {

font-size: 16px;

}

}

#### Schaalbare afbeeldingen

Bij responsive websites horen natuurlijk ook responsive afbeeldingen.

Je kande afbeelding een breedte van 200, 400 of 600 pixels te geven. Alleen is dat natuurlijk niet responsive.

Dit zorgt ervoor dat afbeeldingen altijd 100% van de breedte van hun parent-element aannemen.

img {

width: 100%;

}

Houdt er wel rekening mee dat bij afbeeldingstypes zoals avatars (kleine profiel-foto's) of icoontjes wel een vaste breedte en/of hoogte wil aanhouden.

de flex-basis-, flex-grow- en flex-shrink-properties. De flex-basis-property werkt niet op afbeeldingen, enkel op normale elementen.

losop door de afbeelding in een wrapper-element te wikkelen, zoals bijvoorbeeld een <span>:

.large-image {

width: 100%;

max-width: 600px;

}

img {

width: 100%;

}

.image-wrapper {

flex-basis: 600px;

flex-grow: 0;

flex-shrink: 1;

}

##### <picture>

The <picture> tag gives web developers more flexibility in specifying image resources.

The most common use of the <picture> element will be for art direction in responsive designs. Instead of having one image that is scaled up or down based on the viewport width, multiple images can be designed to more nicely fill the browser viewport.

<p>Resize the browser window to load different images.</p>

<picture>

<source media="(min-width:650px)" srcset="img\_pink\_flowers.jpg">

<source media="(min-width:465px)" srcset="img\_white\_flower.jpg">

<img src="img\_orange\_flowers.jpg" alt="Flowers" style="width:auto;">

</picture>

https://css-tricks.com/a-guide-to-the-responsive-images-syntax-in-html/

#### Schaalbare eenheden

Pixels zijn een **absolute maat**. Pixels zijn namelijk niet afhankelijk van andere elementen:

relatieve maten:

* % relatief aan het parent-element waar het in gebruikt wordt;
* vw relatief aan de breedte van de huidige viewport;viewwidth
* vh relatief aan de hoogte van de huidige viewport; // viewheight
* em relatief aan het parent-element waar het in gebruikt wordt;
  + Een **em** is in de typografie een lengtemaat. Omdat de breedte van deze letter afhangt van het in gebruik zijnde lettertype is het geen absolute afstandsmaat.
* rem relatief aan het root-element (<html>-tag) van het document;
  + In CSS staat "rem" voor een relatieve eenheid van de lettergrootte die gebaseerd is op de lettergrootte van het root element (meestal **<html>**). "Rem" staat voor "root em".

##### standaard

* Afbeelding met tekst, Lettertype, schermopname, nummer

  Automatisch gegenereerde beschrijving

##### Aangepast

Html {

font-size: 12px;

}

Deze standaardwaarde vertaalt zich dan als volgt:

* Afbeelding met tekst, Lettertype, schermopname, nummer

  Automatisch gegenereerde beschrijving
* REM is gebaseerd op de instelling van het root HTML-element. Iedere afmeting die je uitdrukt in REM, neemt de HTML-rootgrootte als basis voor de berekening. 2rem betekent overal op de pagina 24 pixels.
* EM, daarentegen, is gebaseerd op de font-size van het *bovenliggende parent-element*. Dit kan per element verschillen! 2em betekent in een parent met een font-size van 16px iets anders dan in een parent met een font-size van 24px.

<section class="normal-parent">

<h1>Deze titel is <code>24px</code> groot...</h1>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit...</p>

</section>

<section class="bigger-parent">

<h1>Deze titel is <code>48px</code> groot!</h1>

<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit....</p>

</section>

html {

font-size: 12px;

}

h1 {

font-size: 2em;

}

p {

font-size: 1em;

}

.bigger-parent {

font-size: 2em;

}

#### Waarom zijn relatieve maten dan handig?

Relatieve maten zorgen ervoor dat we onze webpagina op meerdere vlakken - niet alleen de lettergrootte - netjes kunnen laten op- en afschalen. Hierin kunnen we ook nog eens inspelen op de verschillende screen width.

html {

/\* voor mobiel \*/

font-size: 16px;

}

@media screen and (min-width: 900px) {

/\* voor tablets \*/

html {

font-size: 18px;

}

}

@media screen and (min-width: 1200px) {

/\* voor desktops en groter \*/

html {

font-size: 20px;

}

}

Hiermee zeggen we: "De standaard font-grootte voor mobiele apparaten is 16 pixels, op tablets is dat 18 pixels en op desktops is dat 20 pixels." Wanneer we niet alleen de groottes van onze letters, maar ook onze paddings en margins vanaf nu in EM of REM declareren, zorgt dit ervoor dat ook de *verhoudingen* tussen elementen beter mee-schalen.

Zo wordt in onderstaand voorbeeld niet alleen de tekst, maar ook de ruimte rondom de tekst groter naarmate het scherm breder wordt:

h1 {

font-size: 3rem;

}

header {

background-color: #ffbab9;

text-align: center;

padding: 8rem 0;

}

* Gebruik EM als het belangrijk is dat properties als padding, margin en border mee schalen met de tekstgrootte van dit specifieke element;
* Voor alle andere situaties: gebruik REM!

#### Syntax

A media query is composed of an optional media type and any number of media feature expressions, which may optionally be combined in various ways using logical operators. Media queries are case-insensitive.

Media types define the broad category of device for which the media query applies: all, print, screen. The type is optional (assumed to be all) except when using the not or only logical operators.

* Media features describe a specific characteristic of the user agent, output device, or environment:
* any-hover
* any-pointer
* aspect-ratio
* color
* color-gamut
* color-index
* device-aspect-ratio Deprecated
* device-height Deprecated
* device-width Deprecated
* display-mode
* dynamic-range
* forced-colors
* grid
* height
* hover
* inverted-colors
* monochrome
* orientation
* overflow-block
* overflow-inline
* pointer
* prefers-color-scheme
* prefers-contrast
* prefers-reduced-motion
* prefers-reduced-transparency Experimental
* resolution
* scripting
* update
* video-dynamic-range
* width.