

HAND
BOEK

HTML5 EN CSS3

Peter Doolaard

5^E EDITIE

INCLUSIEF GRATIS
WEBVERSIE VAN
HET BOEK



handboek

HTML5 & CSS3

Vijfde editie

ISBN: 978-94-6356-081-8

NUR: 994

Trefw.: HTML, CSS, webstandaarden, webdesign, internet

Omslag: Artifex Graphics, Roosendaal

Vormgeving (concept): Aad Metz, De Meern

Opmaak: Peter Doolaard redactie en vormgeving, Schoonhoven

Druk: Bariet Ten Brink, Meppel

Eerste druk: april 2019

Dit boek is gedrukt op een papiersoort die niet met chloorhoudende chemicaliën is gebleekt. Hierdoor is de productie van dit boek minder belastend voor het milieu.

Copyright © 2019 Van Duuren Media B.V.

Van Duuren Informatica is een imprint van Van Duuren Media B.V.

Alle rechten voorbehouden. Niets uit deze uitgave mag worden verveelvoudigd, opgeslagen in een geautomatiseerd gegevensbestand, of openbaar gemaakt, in enige vorm of op enige wijze, hetzij elektronisch, mechanisch, door fotokopieën, opnamen, of enige andere manier, zonder voorafgaande toestemming van de uitgever.

Voorzover het maken van kopieën uit deze uitgave is toegestaan op grond van artikel 16B Auteurswet 1912 j" het Besluit van 20 juni 1974, St.b. 351, zoals gewijzigd bij Besluit van 23 augustus 1985, St.b. 471 en artikel 17 Auteurswet 1912, dient men de daarvoor wettelijk verschuldigde vergoedingen te voldoen aan de Stichting Reprorecht. Voor het overnemen van gedeelte(n) uit deze uitgave in bloemlezingen, readers en andere compilatie- of andere werken (artikel 16 Auteurswet 1912), in welke vorm dan ook, dient men zich tot de uitgever te wenden.

Ondanks alle aan de samenstelling van dit boek bestede zorg kan noch de redactie, noch de auteur, noch de uitgever aansprakelijkheid aanvaarden voor schade die het gevolg is van enige fout in deze uitgave.

Registreer uw boek!

Van Duuren Media biedt haar lezers een unieke service: door uw boek op onze website te registreren, krijgt u gratis toegang tot extra content:

- U kunt de webversie van het boek lezen en doorzoeken in uw browser op computer of tablet (niet via een e-reader).
- U krijgt het e-zine Computer Creatief in uw inbox. Dit digitale magazine bevat artikelen met tips en nieuws over computer & tablet, fotografie & video, webdesign & internet.
- U ontvangt regelmatig onze nieuwsbrief in uw mailbox met relevante informatie over nieuwe boeken, blogs, boekpresentaties en interessante kortingsacties.

Om te registreren:

- 1 Ga naar www.vanduurenmedia.nl/registreren.
- 2 Vul het formulier in en gebruik de registratiecode RkY2Mc79Tq

Blog Computer Creatief



Het blog Computer Creatief is een actuele aanvulling op onze boeken. Het blog geeft de mogelijkheid om actueler, breder en dynamischer te informeren over de nieuwste ontwikkelingen op het gebied van hard- en software, fotografie en camera's. Het blog vindt u op www.computercreatief.nl.

Sociale media

Volg ons ook online! Via onze socialemediakanalen blijft u dagelijks op de hoogte van nieuwe boeken, interviews en recensies.



Volg ons via @VanDuurenMedia



Geen link provided



Zie [linkedin.com/company/van-duuren-media](https://www.linkedin.com/company/van-duuren-media)



Ontdek de voordelen van digitaal lezen op Yindo

Op Yindo vindt u digitale boeken van verschillende uitgeverijen, waaronder Van Duuren Media; deze zijn tegen betaling in te zien. Enkele voordelen van Yindo:

- U kunt uw boeken permanent of tijdelijk aanschaffen.
- Met de zoekfunctie doorzoekt u een boek op onderwerp of trefwoord.
- U leest de boeken op uw computer – geen e-reader vereist.
- Het aanbod omvat leverbare en niet meer leverbare boeken.

Kijk voor meer informatie op www.vanduurenmedia.nl/yindo

Voorwoord

Wie geïnteresseerd is in webdevelopment en een beetje volgt wat daar zoal over wordt geschreven, kan de indruk krijgen dat 't het domein is geworden van programmeurs. Angular, React, Vue.js, het zijn maar enkele van de vele JavaScript-frameworks die worden gebruikt om webapps en websites te bouwen. Wie JavaScript niet beheert, doet niet mee.

Laat me u geruststellen, zelfs met een JavaScript-framework valt het zonder gedegen kennis van HTML en CSS niet mee om een degelijke website te bouwen. Degelijk in de zin van goed gestructureerd, zonder overbodige code, bruikbaar voor mensen met beperkingen en zo vormgegeven dat aanpassingen zonder veel moeite mogelijk zijn. Die kennis vindt u in dit boek.

HTML en CSS zijn volwassen webtalen, die bovendien nog volop in ontwikkeling zijn. HTML maakte de grote sprong vooruit met de komst van HTML 5, vooral door nieuwe, betekenisvolle structuur elementen en handige invoer elementen voor formulieren. HTML is intussen toe aan versie 5.3.

CSS zit sinds de introductie van CSS3 in een groeistuip waarvan het eind nog niet in zicht is. Nieuwe mogelijkheden voor opmaak blijven maar komen, met als recente hoogtepunten robuuste technieken voor het opmaken van complete pagina's (gridlay-out) en componenten (flexbox). Maar ook typografie wordt steeds verfijnder, net als de mogelijkheden voor uitlijning en het werken met kleuren en vormen. Vormgeving met CSS is niet makkelijker geworden, maar geeft wel steeds meer voldoening.

Bent u over 500 pagina's klaar met leren? Oh nee, lang niet. Sorry. Dit boek leert u veel, maar kan onmogelijk elk onderwerp volledig uitdiepen. U beschikt wel over brede basiskennis, verdiept door analyses van (praktijk) voorbeelden. Bovendien leert u niet alleen dat dingen werken, maar vooral waarom ze werken.

En wilt u all-round worden in webdevelopment, leer dan ook programmeren met JavaScript.

Ik wens u veel plezier bij het leren en bouwen!

Peter Doolaard
Schoonhoven, april 2019

reacties@handboek-html-css.nl

Inhoud

1	Webtalen, browsers en editors	1
	Introductie	2
	Een korte terugblik	2
	Afspraken over webtalen	3
	Twee smaken HTML	5
	Wat HTML 5 is	6
	Kennismaken met HTML en CSS	8
	Aan de serverkant	8
	De functie van HTML	9
	De functie van CSS	10
	Browsers	12
	Google Chrome	13
	Microsoft Edge (Internet Explorer)	14
	Mozilla Firefox	15
	MacOS: Safari	16
	HTML-editors	16
	Visual Studio Code	17
	Commerciële editors/IDE's	19
	Aanvullende hulpmiddelen	19
	Hulpmiddelen voor validatie	19
	Hulpmiddelen voor ontwikkeling	20
	Over dit boek	25
	Oefenbestanden	25
	Samenvatting	25
	Oefeningen	26
2	De basis van HTML	27
	Kenmerken van HTML-documenten	28
	Elementen, tags en tekst	28
	Opbouw van een element	30
	Elementen nesten	31

Kleine letters	32
Sluiten in de goede volgorde	32
De HTML-code van webpagina's bekijken	33
Attributen	36
Kenmerken van attributen	36
Globale attributen	37
Microdata	42
WAI-ARIA	42
Kenmerken van HTML 5	43
Omringende technieken	45
De basis van een HTML 5-document	46
<!DOCTYPE html>	47
<html lang="nl">	47
<head>	48
<meta charset="utf-8">	48
<title>	48
<link>	49
<body>	49
</html>	50
Paginastructuur	50
Document Object Model – HTML DOM	50
Het contentmodel van HTML 5	51
Secties markeren	54
<article> (sectioning content)	55
<section> (sectioning content)	56
<nav> (sectioning content)	57
<aside> (sectioning content)	58
Koppen: <h1> t/m <h6>	59
<header> (flow content)	60
<footer> (flow content)	61
Samenvatting	63
Oefeningen	63
 3 Tekst markeren	65
 Inleiding	66
Koppen in HTML-documenten	66
Het getal bepaalt de keus, niet de tekstgrootte	66
Titel website is niet automatisch <h1>	67
Koppen markeren	67
Koppen in een header	68
Alineatekst markeren	69
Alinea's: het element <p>	70
Het regeleinde 	71
Afbreken van woorden	72

Speciale betekenis aangeven	72
Klemtoon leggen met <code></code>	73
Nadruk leggen met <code></code>	73
Aanvullende informatie is <code><small></code>	73
Orgeldig maken met <code><s></code>	73
Verwijzing naar boek of film: <code><cite></code> en <code><q></code>	73
Aftortingen en definities: <code><abbr></code> en <code><dfn></code>	74
Programmacode en dergelijke: <code><code></code> , <code><samp></code> en <code><kbd></code>	75
Subscript en superscript: <code><sub></code> en <code><sup></code>	75
Tekst markeren met <code><mark></code>	75
Het containerelement <code></code>	76
Toegevoegd of verwijderd: <code><ins></code> en <code></code>	76
Inhoud groeperen	76
<code><address></code>	76
<code><main></code>	77
Vooraf opgemaakte inhoud: <code><pre></code>	78
Citaten gebruiken: <code><blockquote></code>	79
Lijsten maken: <code></code> , <code></code> en <code></code>	79
Illustraties en bijschriften: <code><figure></code> en <code><figcaption></code>	84
FAQ, metadata en definities: <code><dl></code> , <code><dt></code> , <code><dd></code>	88
Nieuw thema markeren: <code><hr></code>	89
Het laatste redmiddel: <code><div></code>	90
Speciale tekens in webpagina's	90
Van ASCII naar UTF-8	91
Referenties gebruiken	92
Samenvatting	92
Oefeningen	93
 4 Koppelingen maken	95
 Inleiding	96
Verbindingen leggen met <code><a></code>	97
Attributen van <code><a></code>	98
Bladwijzers maken	101
Link naar een grote foto	102
Link naar het vervolg van een artikel	103
Link naar een e-mailadres	104
Link naar een telefoonnummer	104
Een pdf-bestand weergeven of downloaden	105
Klik hier voor richtlijnen	105
Externe bestanden koppelen met <code><link></code>	108
Stylesheet koppelen	108
Een favicon gebruiken	109
Opeenvolgende pagina's	110
Vooraf laden: prefetch, preload, prerender en dns-prefetch	111

Knoppen en hyperlinks	113
Samenvatting	114
Oefeningen	114
5 Beeld, geluid en andere externe inhoud	115
Inleiding	116
URL's en structuur van de website	116
Lokale site	118
Soorten afbeeldingen	119
Bitmaps	119
Vectorafbeeldingen: SVG	120
Bestandsformaten voor bitmaps	122
GIF	122
JPEG	123
PNG	124
WebP	125
De juiste afbeelding voor het scherm	125
Afbeeldingen plaatsen met 	126
De bron: src	127
Beschrijvende tekst: alt	127
Afmetingen: width en height	128
Klikbare gebieden: usemap	129
Responsive images	130
Srcset voor schermen met hoge resolutie	130
Srcset voor variabele afmetingen	131
Bijgesneden afbeeldingen: <picture>	133
Het type afbeelding selecteren	134
Het verschil tussen en <picture>	134
Externe HTML-inhoud: <iframe>	135
Attributen van <iframe>	135
Beveiliging	137
Insluiten met <embed> of <object>	138
Video in een webpagina	138
Bestandstypen voor video	139
Een codec kiezen	139
Video coderen	140
Het element <video>	140
De bron: <source>	141
Ondertiteling met <track>	142
Titelbestanden: WebVTT	144
Audio op de website gebruiken	145
Samenvatting	146
Oefeningen	147

6 Formulieren maken	149
De rol van formulieren	150
Formulieren aan de clientkant	150
De rol van de webserver	151
Een compleet formulier	152
De basis van een formulier: <form>	156
Buiten <form> mag ook	156
De verwerking: action	156
Verzendwijze: method	156
Codering: enctype	157
Automatisch aanvullen: autocomplete	157
Niet valideren: novalidate	158
Direct invoeren: autofocus	159
De naam: name	159
Het doel: target	159
Buitenspelelementen: form	159
Labels bij invoervelden	161
De tabvolgorde	162
Het element <input>	162
Attributen van <input>	163
Soorten invoer: het attribuut type	170
Telefoonnummers: type="tel"	170
E-mailadressen: type="email"	171
Webadressen: type="url"	171
Wachtwoorden: type="password"	171
Getallen: type="number"	172
Een bereik: type="range"	172
Kleuren: type="color"	173
Datum en tijd	174
Selectievakjes: type="checkbox"	175
Keuzerondjes: type="radio"	176
Bestanden uploaden: <type="file">	178
Verzendknop: type="submit"	178
Resetknop: type="reset"	179
Afbeeldingknop: type="image"	179
Functieloze knop: type="button"	180
Verborgen waarden: type="hidden"	180
Het element <button>	180
Kiezen uit een lijst: <select>, <datalist>, <option>	181
<option>	182
<select>	183
<datalist>	184
Uitgebreide tekstinvoer: <textarea>	186

Groeperen met <fieldset> en <legend>	186
Voortgang tonen: <progress>	189
Schaal: <meter>	190
Resultaat: <output>	191
Samenvatting	192
Oefeningen	193
 7 Tabellen maken	195
Inleiding	196
De structuur van een HTML-tabel	196
De basis: <table>	198
Rijen en kolommen: <tr> en <td>	198
Koptekst: <thead>	199
Een bijschrift: <caption>	200
Overige elementen	201
De tabelinhoud: <tbody>	201
Voettekst: <tfoot>	201
Kolomgroepen: <colgroup> en <col>	202
Voorbeelden van tabellen	203
Samenvatting	206
Oefeningen	207
 8 De basis van CSS	209
Wat CSS is	210
Voorgeschiedenis	210
En toen was er CSS3	211
CSS3 is gewoon CSS	211
Waarom CSS zo handig is	213
De taal CSS	213
Waarden en eenheden	215
Globale waarden	217
Lengte	217
De pixel	217
De eenheid em	218
De makkelijke em; rem	218
Handig voor tekstvakken: ch	219
Enheden gebaseerd op de viewport	219
Overige eenheden	220
Samenvatting afmetingen	220
CSS-verwerking door de browser	221
De boomstructuur	221
Waarden toewijzen	222

De cascade	223
Specificiteit berekenen	224
Overerving	226
Percentages werken door	227
De plaats van de CSS-declaraties	228
Het attribuut style	229
Het element <style>	229
Extern bestand: <link>	230
Het CSS-bestand	231
Commentaar toevoegen	231
Indeling van CSS-bestanden	232
Andere stijlbladen importen	232
De ingebouwde stijlen aanpassen	232
Elementen benaderen met selectors	234
Elementnaam, klasse of beide?	235
Elementnamen en selectors	235
Klassen	237
Van alles wat	238
En de winnaar is...	239
Basisselectors	239
Typeselector	240
Universele selector: *	240
Klasseselector	242
ID-selectors	243
Attribuutselectors	243
Pseudoklassen	244
Pseudo-elementen	248
Pseudo-elementen level 4	250
Combinatieselectors: afstamming, kind en sibling	251
Geen combinator: komma	251
Algemene afstamming: spatie	251
Kindselector	252
Aangrenzend: +	252
Op hetzelfde niveau: ~	252
Best practices	253
Samenvatting	254
Oefeningen	255
 9 Lay-out: boxmodel en weergavemodel	257
 De opbouw van pagina's	258
Browserstijlen zijn er niet voor niets	258
Het boxmodel	259
De eigenschap box-sizing	260

Inhoud

Waarden voor breedte en hoogte	263
Begrensde breedte en hoogte	264
Marges	265
Horizontaal centreren	265
Verticaal centreren	266
Negatieve marge	267
Samengevoegde marges	267
Padding	268
Randen	269
Randdikte	269
Randkleur	269
Transparante rand	270
Randstijl	271
Korte notatie voor randen	272
Overlopende inhoud: overflow	273
Overloop van de scroll	275
De toekomst: logische eigenschappen	276
Breedte en hoogte	277
Het einde van top, bottom, left, right?	277
Wat is het nut?	278
Praktische beperking	279
De toekomst: box alignment	279
Weergavemodel	280
Achter de schermen	281
De eigenschap display	281
Waarden voor binnen en buiten	281
Inhoud (niet) weergeven: visibility	285
Niet verbergen voor screenreaders	285
Samenvatting	286
Oefeningen	287
10 Lay-outs maken met CSS	289
Inleiding	290
Responsive design	290
Media queries	291
Schermresolutie	294
Portret of landschap	294
Beeldverhouding	295
Weergave op mobiele schermen: de metatag viewport	295
CSS-pixels er hardwarepixels (device pixels)	297
Lay-out: positionering	297
Het omvattende blok	297
Positioneringsschema's	298

De eigenschap position	298
Verschuiven met top, right, bottom en left	299
Verschuiven met logische waarden	299
Position: static	299
Position: relative	299
Position: absolute	300
Position: fixed	302
Position: sticky	303
Float en clear	305
Float als hulpmiddel voor lay-outs	307
Block formatting context	310
Lay-out: multi-column	312
Kolommen instellen	312
Marges worden niet samengevoegd	314
Kolommen overspannen	315
Ruimte tussen de kolommen	316
Lijn tussen de kolommen	317
Kolommen vullen	318
Afbreken in kolommen	318
Wanneer multi-column (niet) nuttig is	321
Toepassing: bol.com	322
Toepassing: portfolio	324
Lay-out in een raster: gridlay-out	326
Kenmerken van gridlay-out	327
Hulpmiddelen voor gridlay-out en flexbox	328
Een raster maken	330
De functie repeat()	331
De eenheid fr	331
Rasteritems plaatsen op lijnnummers	332
Kolommen en rijen samen in grid-area	335
De lijnen een naam geven	336
Items plaatsen in gebieden met een naam	338
Afmetingen van tracks instellen	340
Ruimte tussen tracks	341
Het raster zichzelf laten bouwen	342
Het verschil tussen auto-fill en auto-fit	345
Van richting veranderen met grid-auto-flow	345
De gaten in het raster opvullen	346
Uitlijnen in gridlay-out	348
Toepassing: nos.nl	350
Lay-out: flexbox	359
Kenmerken van flexbox	359
Het assenstelsel van flexbox	359
Flex-direction	360

Flex-wrap	361
Flex-flow	362
Flexcontainers en flexitems	362
Items flexibel maken met de eigenschap flex	363
Instellingen voor flex	364
Samenvatting: het formaat van flexitems	365
Items uitlijnen met automarge	366
Uitlijnen op de hoofdas: justify-content	367
Uitlijnen op de kruisassen: align-self	369
Regels uitlijnen: align-content	370
Toepassing: navigatiebalk	371
Toepassing: kaarten	374
Toepassing: nos.nl	378
Samenvatting	380
Oefeningen	381
11 Tekst en typografie	385
Inleiding	386
Lettertype: van systeem of online?	386
Web fonts	387
Lettertypen downloaden: @font-face	389
Eigenschappen van @font-face	390
Een embedcode gebruiken	393
Eigenschappen voor lettertype: font	395
Lettertype	395
Lettergrootte: waarden en overwegingen	396
Geavanceerd: responsieve tekst	397
Toepassing van font-size	398
Grootte aanpassen: font-size-adjust	399
Regelhoogte: line-height	399
Letterstijl: font-style	401
Lettergewicht: font-weight	401
De verzameleigenschap font	402
Uitrekkken of indrukken: font-stretch	403
Kleininkapitaal: font-variant	404
Eigenschappen voor tekst	405
Uitlijnen: text-align en text-align-last	405
Lijneffecten op tekst	405
Inspringen	407
Rechtopstaande tekst: text-orientation	408
Overlopende tekst: text-overflow	409
Tekst met schaduw	409
Tekst omzetten: text-transform	411

Witruimte behouden: white-space	411
Ruimte tussen woorden en letters	412
Afbreken: word-break en overflow-wrap	413
Opmaak van lijsten	414
Opsommingstekens	415
Lijst als navigatie	416
Samenvatting	422
Oefeningen	423
12 Kleur, randen en achtergronden	425
Inleiding	426
Kleurwaarden	426
RGB en RGBA	427
HSL en HSLA	428
Transparantie	430
Kleur van tekst	430
De achtergrond	431
Achtergrondkleur	431
Achtergrondafbeelding	431
background-image	432
background-repeat	432
background-attachment	433
background-position	433
background-clip	435
background-origin	436
background-size	437
background	438
Verschillende achtergrondafbeeldingen	439
Kleurverloop als achtergrond	440
Kleurverlopen	441
Lineair verloop	441
Radiaal verloop	443
Repeterende verlopen	444
Afgeronde hoeken	445
Randafbeelding	447
border-image-source	448
border-image-slice	448
border-image-width	448
border-image-outset	449
border-image-repeat	449
Voorbeeld van een randafbeelding	449
Schaduw	450
De werking van schaduw	451

Inhoud

Material design	452
Afbeeldingen uitsnijden met clip-path	453
Toepassing: header met schuine kanten	456
Beeldfilters	458
Kleuren mengen	458
Isolation	460
Samenvatting	460
Oefeningen	461
 13 Overgangen en animaties, calc(), attr() en CSS-variabelen	463
 Beweging in webpagina's	464
Overgangen (transitions)	464
Kant-en-klare overgangen	467
Positie aanpassen	467
Animatie	469
Opbouw een animatie	469
Keyframes	471
Transformatie	473
Tweedimensionale transformaties	474
Schalend menu	476
Driedimensionale transformaties	478
Perspectief	479
Kijkrichting	480
3D-ruimte	480
Rekenen met calc()	484
Voorbelder	485
CSS-variabelen: de functie var()	485
Kenmerken van variabelen	486
Variabelen declareren	487
Toepassingen van variabelen	487
Attributen uitlezen met attr()	489
All reset alles, behalve variabelen	491
Samenvatting	491
Oefeningen	492

HTML is de belangrijkste taal op internet. Sinds 2011 werken we met HTML 5 en daar wordt nog steeds aan gesleuteld. Dit hoofdstuk beschrijft het ontstaan van HTML 5 en de actuele situatie. Er wordt kort uitgelegd wat HTML en CSS zijn en wat HTML 5 is. Er is een overzicht van browsers en hulpmiddelen voor het schrijven van HTML-documenten en u maakt kennis met handige gereedschappen.

U leert in dit hoofdstuk:

Hoe we hier gekomen zijn.

De rol van HTML en van CSS, en wat HTML 5 is.

Welke browsers niet mogen ontbreken in uw gereedschapskist.

Handige gereedschappen: editors, validators en hulpmiddelen voor ontwikkelaars.

Introductie

Het world wide web met zijn blogs, games, webwinkels, videosites, sociale media, magazines en al die andere soorten sites en apps zou niet bestaan zonder de taal HTML, de afkorting van Hypertext Markup Language. Het fraaie uiterlijk wordt gemaakt met CSS (Cascading Style Sheets) en de logica wordt geprogrammeerd met JavaScript, maar zonder HTML was er niets.

Dit boek gaat over HTML 5. Bij de publicatie van deze editie is de laatste voltooide versie HTML 5.2 uit december 2017 en wordt er gewerkt aan HTML 5.3.

HTML 5 kan niet los worden gezien van de voorgangers, HTML 4.01 en XHTML. HTML 5 is daar een voortzetting van, waarbij onderdelen zijn aangepast, toegevoegd en afgeschaft, maar alle oude webpagina's blijven werken. Afschaffen betekent in dit verband dan ook dat auteurs worden aangemoedigd nieuwe webpagina's volgens nieuwe standaarden te coderen en oude pagina's te moderniseren. Doen ze dat niet, dan blijven de webpagina's toch toegankelijk voor bezoekers. Ze zien die er misschien niet meer zo uit als in 2000 werd bedoeld, maar de browser moet in elk geval de inhoud weergeven.

Een korte terugblik

Toen de eerste editie van dit boek verscheen – voorjaar 2011 – was de inkt van het logo van HTML 5 nog nat. De specificatie was nog volop in ontwikkeling, maar de belangrijkste elementen waren al omarmd door de gemeenschap van webbouwers en door browserfabrikanten. In 2014 heeft de beheerder van de webtalen, het World Wide Web Consortium (W3C) de specificatie ‘definitief’ vastgesteld (in W3C-termen is het dan de aanbevolen specificatie, gelabeld als *recommendation*). Definitief staat niet voor niets tussen aanhalingsstekens, want het werk aan de specificatie gaat gewoon door. Het betekent alleen dat die versie wordt bevroren en dat het werk verder gaat in de volgende versie. In november 2016 werd HTML 5.1 de aanbevolen versie en in oktober 2017 werd dat HTML 5.1.2nd edition. Snel daarna, in december 2017, werd HTML 5.2 de aanbevolen versie en is het werk aan HTML 5.3 begonnen. Bent u er nog?

Elke revisie bevat kleine veranderingen. Vaak zijn het aanvullingen en verbeteringen gebaseerd op hoe HTML wordt gebruikt en op reacties uit de gebruikersgemeenschap. Soms vervalt een element (omdat het nauwelijks wordt gebruikt) of verandert de toepassing ervan. *Breaking changes* zijn in elk geval niet verwachten, omdat het web nu eenmaal niet mag ‘breken’.



Afbeelding 1.1 Het logo van HTML5, in 2011 gepresenteerd door het W3C.

Naast de versie van het W3C is er een tweede HTML-specificatie en die gaat door het leven als *HTML Living Standard*. Een updatedatum geeft aan wanneer die voor het laatst is bijgewerkt. Hierbij zijn er geen bevroren versies, het is een ‘levende’ standaard waar continu aan wordt gewerkt.



Afbeelding 1.2 HTML Living Standard, beheerd door WHATWG.

Dat er twee standaarden bestaan klinkt misschien vreemd (en dat is het ook), maar het komt doordat de ontwikkeling van HTML een tijdje heeft stilgelegen.

Afspraken over webtalen

Sinds 1994 wordt geprobeerd lijn te brengen in de talen waarmee websites (en andere webtoepassingen) worden gemaakt. Die talen zijn onder meer Hyper-text Markup Language (HTML) en Cascading Style Sheets (CSS). Een standaard voor deze talen is geen overbodige luxe. Toen de belangstelling voor internet groeide en er meer browsers op de markt werden gebracht, deed elke fabrikant wat hem leuk leek. De HTML-standaard werd wel ingebouwd, maar daar werd een eigen draai aangegeven. Zo ontstond een situatie waarin browsers naast standaard-HTML ook eigen tags hadden (berucht waren `<blink>` voor knipperende tekst en `<bgsound>` voor achtergrondgeluid). Bij CSS was de situatie vergelijkbaar. Zo kon een website met dezelfde code er in elke browser anders uitzien, anders werken of compleet uit elkaar vallen. Vooral Microsoft heeft met eigen interpretaties van de standaarden in Internet Explorer ontwikkelaars tot wanhoop gecreven. Intussen zijn dergelijke problemen grotendeels opgelost en werken de talen in alle browsers vrijwel hetzelfde.

Voor het beheer van de talen is het World Wide Web Consortium opgericht, kortweg W3C. In verschillende werkgroepen wordt overlegd door browserfabrikanten zoals Google, Microsoft, Mozilla en Opera. Omdat W3C ook allerlei andere webstandaarden beheert, vindt u onder de leden ook bedrijven als Bloomberg, Vrije Universiteit Amsterdam, Nokia en Walt Disney Company. Het doel van bijvoorbeeld de werkgroep Web Platform is afspraken maken over hoe een auteur in een webdocument aangeeft dat iets een hyperlink, een kop of een opsomming is, en hoe een browser daarmee zou moeten omgaan.



World Wide Web Consortium

Kijk voor meer informatie over wat het W3C is en doet op w3.org. Dat is de thuishaven van deze in Amerika gevestigde organisatie. U vindt er informatie over elke webstandaard. Daar hoort JavaScript niet bij, dat wordt beheerd door Ecma (www.ecma-international.org).

The screenshot shows the W3C website's 'ALL STANDARDS AND DRAFTS' page. At the top, there's a search bar and navigation links for 'STANDARDS', 'PARTICIPATE', 'MEMBERSHIP', and 'ABOUT W3C'. Below that is a filter section with dropdowns for 'Type', 'Status', 'Topic', 'Version', and a 'Reset' button. The main area displays a grid of standard cards. Some visible cards include:

- The Screen Orientation API**: Last updated: 16 Dec 2010. Status: Working Draft. Participants: Henning Kuehl, ...
- Trace Context Protocol Registry**: Last updated: 16 Dec 2010. Status: Working Draft. Participants: ...
- Payment Method: Basic Card**: Last updated: 16 Dec 2010. Status: Working Draft. Participants: ...
- User Timing Level 3**: Last updated: 16 Dec 2010. Status: Working Draft. Participants: ...
- Navigation Timing Level 2**: Last updated: 16 Dec 2010. Status: Working Draft. Participants: ...
- Generic Sensor API**: Last updated: 16 Dec 2010. Status: Working Draft. Participants: ...
- Server Timing**: Last updated: 16 Dec 2010. Status: Working Draft. Participants: ...
- Ambient Light Sensor**: Last updated: 16 Dec 2010. Status: Working Draft. Participants: ...
- Gyroscope**: Last updated: 16 Dec 2010. Status: Working Draft. Participants: ...

Afbeelding 1.3 Een (heel) klein deel van de standaarden en concepten op de website van W3C.

Het is allerminst vanzelfsprekend dat er nu een W3C-standaard van HTML 5 is. Eind jaren negentig, de standaard HTML 4.01 was klaar, besloot het W3C zich volledig te richten op XHTML, een vorm van HTML gemodelleerd naar de strikte regels van XML. Het begon met het herschrijven van HTML naar XML (XHTML 1.0) en tegelijk werd gewerkt aan een geheel nieuwe versie (XHTML 2)

die niet uitwisselbaar zou zijn met eerdere HTML- en XHTML-talen. Er was daarmee een route gekozen waarbij het bestaande web zou blijven zoals het was (HTML 4) en er daarnaast een niet-uitwisselbaar nieuw web zou ontstaan met op XML gebaseerde applicaties.

Tijdens dat ontwikkelproces zagen Mozilla en Opera in 2004 nieuwe mogelijkheden voor HTML en zij hebben voorgesteld om toch door te gaan met de ontwikkeling van HTML, maar daar waren bij het W3C niet genoeg medestanders voor. In reactie daarop besloten Apple, Mozilla en Opera dan maar zelf aan de slag te gaan met HTML 5. Daarvoor richtten zij in 2004 de Web Hypertext Application Technology Working Group op, kortweg WHATWG. Hun voornaamste uitgangspunt was dat webontwikkelaars een webtaal gebaseerd op het oude HTML 4 tot hun beschikking zouden krijgen, dus uitwisselbaar, maar met nieuwe mogelijkheden om een volwaardig alternatief te kunnen bieden voor native webapps (iOS en Android) en gepatenteerde mediatechnieken zoals Flash van Adobe en Silverlight van Microsoft. Dit initiatief viel bij de gemeenschap van webontwikkelaars in goede aarde. Ontwikkelaars ontvingen de ontwikkelingen met gejuich en browserfabrikanten gingen de nieuwe ideeën inbouwen in hun browsers, en daardoor ging uiteindelijk ook bij het W3C het roer om. De werkgroep XHTML is eind 2009 opgeheven en ook bij het W3C gaat alle aandacht weer uit naar HTML.

Het zijn de gedrevenheid van WHATWG en de snelheid waarmee in browsers nieuwe mogelijkheden worden ingebouwd die de ontwikkeling van HTML in een stroomversnelling hebben gebracht. Google (Chrome), Mozilla (Firefox), Opera en Microsoft (Edge) brengen in hoog tempo updates voor de browser uit. Het resultaat is dat de nieuwste versies van alle toonaangevende browsers veel nieuwe mogelijkheden ondersteunen. Het werkt als een vliegwheel: het feit dat nieuwe opties snel worden opgenomen in browsers stimuleert de ontwikkeling.



Automatische updates

Er worden niet alleen sneller nieuwe versies van browsers uitgebracht, ze worden ook veel sneller op grote schaal gebruikt door het internetpubliek. Dat komt doordat browserupdates automatisch worden gedownload en geïnstalleerd, zonder tussenkomst van de gebruiker.

Twee smaken HTML

Zijn er nu twee smaken HTML? In zekere zin is dat zo. Hoewel de oorspronkelijke HTML 5-aanbeveling op de website van het W3C het resultaat is van samenwerking tussen W3C en WHATWG, hebben de wegen van de groepen zich na 'HTML 5.0' gescheiden. Eerst leek het erop dat W3C HTML 5 helemaal

niet verder zou ontwikkelen, maar intussen weten we beter. WHATWG gaat cok gewoon door en zo lopen er twee HTML-specificaties naast elkaar. Wat is de zin daarvan en vooral: wat moet de webontwikkelaar ermee?

Hoewel W3C en WHATWG van mening verschillen over het omgaan met (web)standaarden, zijn er niet twee totaal verschillende HTML-standaarden. Een webontwikkelaar merkt er nauwelijks iets van. Zijn enige zorg is of een bepaalde optie goed werkt in de browser en het is niet moeilijk om daar achter te komen. (U mag alle websites vergeten, maar onthoud altijd deze twee: caniuse.com en developer.mozilla.org.)

Uiteindelijk profiteren webontwikkelaars en internetgebruikers alleen maar van de situatie. De clubs houden elkaar en de browserfabrikanten scherp, zodat een constante stroom van verbeteringen en nieuwe mogelijkheden in stand wordt gehouden. Dat houdt het web levend en vitaal, en het leven van een webontwikkelaar boeiend.



Bekijk de specificaties

Uiteraard zijn alle specificaties online beschikbaar. De actuele W3C-aanbeveling is te vinden op www.w3.org/TR/html. Werk in uitvoering (het concept van de volgende revisie) is te vinden via de pagina www.w3.org/TR/ met als filters de tag HTML en de status Working Draft. De specificatie van WHATWG is te vinden op html.spec.whatwg.org/multipage/ en toont altijd de versie van de dag.

W3C Working Draft

HTML 5.3

W3C Working Draft, 18 October 2018

This version:

<https://www.w3.org/TR/2018/WD-html53-211018/>

Latest published version:

<https://www.w3.org/TR/html5/>

Editor's Draft:

<https://w3c.github.io/html/>

W3C

Afbeelding 1.4 Hier gaan we het uitgebreid over hebben.

Wat HTML 5 is

HTML 5 is eigenlijk niet meer dan een upgrade van HTML 4. We hebben het dan alleen over de markeertaal HTML. Maar HTML 5 wordt ook gebruikt als verzamelnaam voor webtechnieken om moderne (mobiele) webapplicaties te bouwen. Naast HTML gaat het dan ook over een verzameling definities waar mee de ontwikkelaar toegang krijgt tot onderdelen van de browser of

smartphone (API's, zie de opmerking hierna) en aanvullende technieken zoals CSS, JavaScript, XML, JSON en SVG. Maar deze horen niet bij de HTML 5-standaard zelf. Ze hebben wel een sterke relatie en het is bijna niet mogelijk om moderne websites en webapps te bouwen zonder er een of meer te gebruiken. Op www.w3.org/TR/ kunt u zoeken met de tag Web API om een overzicht te bekijken. Ook op spec.whatwg.org vindt u een overzicht van technologieën voor het webplatform.

Over HTML en CSS leest u alles in dit boek. Voor de andere technieken moet u andere bronnen raadplegen. Een korte toelichting op de functie ervan:

- JavaScript is een programmeertaal (scripttaal) om onder meer webpagina's interactief te maken (onder)delen van de pagina updaten, gegevens lezen uit een databron (JSON), een winkelwagen bijhouden, cookies plaatsen en lezen enzovoort.
- XML staat voor Extensible Markup Language en is een markeringstaal (zoals HTML). XML is ontwikkeld om gestructureerde gegevens uit te wisselen tussen (web)toepassingen op een manier die ook voor mensen leesbaar is.
- JSON is de afkorting van JavaScript Object Notation en is een veelgebruikt alternatief voor XML. De functie is *hetzelfde*: uitwisseling van gegevens tussen een databron en een applicatie. JSON wordt verder niet besproken.
- SVG staat voor Scalable Vector Graphics. Dit is een op XML gebaseerd bestandsformaat voor vectorafbeeldingen (vectorafbeeldingen kunnen worden vergroot of verkleind zonder dat ze rafelig of blokkerig worden zoals bitmapafbeeldingen). Ook SVG valt buiten het bestek van dit boek.



Toegangspunt voor ontwikkelaars: API

De afkorting API staat voor application programming interface. Het is een toegangspunt voor aanvullende functies waarbij de ontwikkelaar eenvoudig toegang heeft tot die functies zonder dat hij precies hoeft te weten ze werken. Hij hoeft alleen te weten hoe hij toegang krijgt. Een vergelijking kan zijn het bedienen van software met een grafische interface of met getypte opdrachten in een terminalvenster. Voordbeelden zijn API's voor het opnemen van beeld en geluid met een telefoon, voor het werken met een muis, pen of aanraakscherm, voor het werken met de geografische locatie van de gebruiker, of voor het opslaan van gegevens op het apparaat van de gebruiker. Hierbij is altijd JavaScript nodig.

Kennismaken met HTML en CSS

Lees zeker de volgende paragrafen als HTML en het maken van webpagina's voor u volledig onbekend terrein zijn. Hebt u al wat ervaring met HTML en CSS, dan kunt u ze eventueel overslaan.

Het eerste deel van dit boek gaat over HTML. Maar tenzij u kale tekstpagina's wilt maken, kunt u niet zonder CSS, wat staat voor Cascading Style Sheets. Deze taal is ontwikkeld om aan te geven hoe de met HTML gemarkeerde teksten (en afbeeldingen en meer) eruit moeten zien en hoe de webpagina moet worden ingedeeld, de lay-out. Daarom gaat het tweede deel van het boek over CSS.

Naast HTML en CSS kunnen programmeertalen worden gebruikt om webpagina's tot leven te brengen. JavaScript is zo'n programmeertaal. Deze wordt bijvoorbeeld gebruikt om acties na een klik op een knop uit te voeren of om onderdelen van een webpagina te manipuleren. Maar ook complete apps worden gebouwd met JavaScript-frameworks zoals Angular, React of Vue.js. Het is belangrijk om u ook in JavaScript te verdiepen als u verder wilt met webontwikkeling.

HTML, CSS en ook JavaScript doen hun werk in de browser op het apparaat waarmee een bezoeker uw pagina's bekijkt. Er is natuurlijk communicatie met de webserver, maar het werk wordt gedaan door de browser. Daarmee vormen uw bezoeker, zijn computer/tablet/telefoon en vooral zijn browser de client-side.

Aan de serverkant

Er zijn ook talen en technieken waarbij het werk op de webserver wordt gedaan, server-side dus, bijvoorbeeld het bij elkaar zoeken van componenten waaruit een webpagina wordt samengesteld. De server stuurt dan een kant-en-klare webpagina naar de client (de browser). Een veelgebruikte programmeertaal op webservers is PHP. Deze taal wordt bijvoorbeeld toegepast om een contentmanagementsysteem (cms) zoals WordPress (www.wordpress.org) of Joomla! (www.joomla.org) aan te sturen. PHP wordt daarbij onder meer gebruikt om pagina's samen te stellen aan de hand van informatie die in een database is opgeslagen. PHP blijft in dit boek verder buiten beschouwing.



Nog meer talen

Er zijn nog veel meer talen voor het web dan de hier genoemde, maar tenzij u specialistische toepassingen wilt ontwerpen, krijgt u daar niet mee te maken. Op w3c.org kunt u er echter van alles over vinden.



JavaScript op de server

Met de opkomst van JavaScript-frameworks zoals Angular, React en Vue.js is ook de rol van JavaScript als servertaal belangrijker geworden. Daarbij spelen het platform Node.js (www.nodejs.org) en de pakketbeheerder npm (www.npmjs.com) een centrale rol. Met Node.js kunnen toepassingen worden geschreven in JavaScript (en andere talen die naar JavaScript kunnen worden omgezet, zoals TypeScript). Node.js bevat onder meer een webserverfunctie, waarmee op de server webpagina's en apps kunnen worden samengesteld uit componenten die zijn gebouwd met de genoemde frameworks (*server side rendering*). Dit in tegenstelling tot pure HTML-pagina's die u kant en klaar op de server zet. In dit boek kunnen al die mogelijkheden niet worden uitgelegd, maar u weet nu dat er na HTML en CSS nog veel meer te ontdekken is.

De functie van HTML

HTML staat voor Hypertext Markup Language. Met HTML worden de *structuur* van de pagina en de *betekenis* van de elementen in die pagina aangegeven. (In Engelstalige bronnen wordt gesproken over *structure* en *semantics*). Er wordt met tags (labels) beschreven wat de kopteksten en alineateksten van een pagina zijn en welke afbeeldingen in de pagina moeten worden geladen, en met hyperlinks wordt aangegeven wat de onderlinge relatie tussen pagina's is. HTML heeft niets te maken met het uiterlijk van een pagina.

HTML is geen programmeertaal. Het is een markeertaal (*markup*). Vandaar ook de afkorting Hypertext Markup Language, oftewel 'een markeertaal voor hypertext'. Hypertext is tekst met hyperlinks (koppelingen) naar andere tekst, afbeeldingen, audio, video en meer.

De hyperlinks, maar ook alle andere onderdelen van een webpagina, zijn de *elementen* van HTML en die worden gemarkerd met tags. Om met HTML te kunnen werken, moet u weten welke elementen er zijn en welke tags daar bij horen. Stel dat u een artikel hebt met een kop en een alinea. De HTML-code hiervoor kan zijn:

```
<article>
  <h1>De functie van HTML</h1>
  <p>HTML staat voor Hypertext Markup Language. Met HTML worden de <i>structuur</i> van de pagina en de <i>betekenis</i> van de elementen in die pagina aangegeven.</p>
</article>
```

Een artikel wordt op een HTML-pagina aangeduid met het element `article`. Het begin van het artikel wordt gemarkerd met de openingstag `<article>`.

Voor de belangrijkste kop op de pagina heeft HTML het element `h1`. Het begin van de koptekst is gemarkeerd met de openingstag `<h1>`. Alles wat hierna volgt (in dit voorbeeld de tekst De functie van HTML) wordt gemarkeerd als koptekst (met de *h* van *header*). De kop loopt door tot de sluittag `</h1>`. Een stuk gemarkeerde tekst wordt altijd afgesloten met een sluittag. Een sluittag heeft dezelfde naam als de openingstag en begint met een voorwaartse slash `</>`.

Hetzelfde geldt voor de alineatekst. De openingstag `<p>` markeert het begin van de alinea (*paragraph*) en de sluittag `</p>` het einde ervan.

In de alinea ziet u nog meer tags, twee keer de openingstag `<i>` en de sluittag `</i>`. Deze markeren belangrijke begrippen. In dit boek zijn die woorden om dezelfde reden cursief. Toch staat `<i>` niet voor *italic* (cursief). ‘Vroeger’, toen HTML-tags ook opmaak markeerden, was dat wel zo, maar nu markeert het element `i` tekst die op de een of andere manier afwijkt van de rest, meer daarover in hoofdstuk 2. De tags `<i>` en `</i>` staan binnen de alineatags `<p>`. Dit insluiten of nesten van tags is een belangrijk principe in HTML. Overigens kunnen niet alle tags worden genest. Hoe dat zit komt in volgende hoofdstukken aan de orde.

Het artikel wordt afgesloten de sluittag `</article>`. Alles wat tussen de begin- en eindtag `<article>` staat, hoort bij het HTML-element `article`. Dat ‘doet’ verder niets, het geeft alleen aan dat alles wat tussen de tags staat bij het artikel hoort. Op die manier is ook duidelijk dat de elementen `h1` en `p` bij elkaar horen. In de browser ziet het voorbeeld er zo uit als in de afbeelding.

HTML bevat meer dan honderd elementen. In dit boek worden de meeste ervan besproken.

De functie van HTML

HTML staat voor Hypertext Markup Language. Met HTML worden de *structuur* van de pagina en de *betekenis* van de elementen in die pagina aangegeven.

Afbeelding 1.5 Een artikel met een kop en een alinea met benadrukte woorden. Het uiterlijk is bepaald door de browser. Deze bevat een basiscopmaak voor elk HTML-element.

De functie van CSS

Met CSS wordt het uiterlijk van de pagina bepaald. CSS-code beschrijft hoe de kopteksten, alineateksten en alle andere elementen op de pagina worden opgemaakt. In CSS wordt aangegeven welk lettertype wordt gebruikt, welke lettergrootte, letterkleur, uitlijning, regelafstand, witruimte tussen onderdelen

en veel meer. Met CSS wordt ook de lay-out van de pagina ingesteld: kolommen, header, footer, kaders enzovoort. Zelfs transformaties, overgangen en animaties zijn mogelijk. CSS heeft dus alles te maken met vormgeving, maar niets met de inhoud van de pagina.

Neem nog even het voorbeeld van het artikel uit de voorgaande paragraaf in gedachten. In HTML wordt aangegeven dat een bepaalde tekst een koptekst moet zijn, maar dat zegt nog niets over het uiterlijk. Welk lettertype en welke kleur moet de koptekst hebben en hoe groot moet die zijn? Dat is de taak van CSS. Onthoud het volgende:

- Met HTML wordt de *structuur* van de webpagina en de *betekenis* van de inhoud gemarkkeerd (kopteksten, tussenkoppen, hyperlinks, alinea's, invulvelden, knoppen en dergelijke).
- Met CSS wordt het *uiterlijk* van de webpagina ingesteld (lettertype, kleur, regelafstand, marges, en dat alles desgewenst per onderdeel van de pagina).

Met een paar regels CSS wordt het voorbeeld uit afbeelding 1.5 opgemaakt:

```
h1 {
    color: hsl(0, 0%, 15%);
    font-family: 'Source Sans Pro', sans-serif;
    font-weight: 600;
}
p {
    color: hsl(0, 0%, 5%);
    font-family: 'Source Serif Pro', serif;
    font-weight: 300;
}
```

De functie van HTML

HTML staat voor Hypertext Markup Language. Met HTML worden de *structuur* van de pagina en de *betekenis* van de elementen in die pagina aangegeven.

Afbeelding 1.6 Hetzelfde artikel als in vorige afbeelding, opgemaakt met CSS. In de HTML-code is niets gewijzigd, alleen het uiterlijk is aangepast met enkele stijlregels.

Het lettertype, het gewicht (de dikte) van de tekst en tekstkleur zijn ingesteld. De tekstgrootte is niet aangepast, maar doordat een ander lettertype is gekozen, is de tekst toch wat groter. Dit zijn eenvoudige aanpassingen en met CSS is heel veel meer mogelijk. De complete CSS-familie bestaat uit enkele honderden eigenschappen die samen alle mogelijke manieren bieden om een webpagina op te maken. Hoe meer CSS-eigenschappen u kunt toepassen, hoe meer mogelijkheden u hebt voor een fraai eindresultaat.



CSS3

Net als de structuurtaal HTML is de opmaaktaal CSS sterk in ontwikkeling. Er wordt al jaren gewerkt aan de nieuwste standaard: CSS3, hoewel die naam eigenlijk niet klopt. Een groot deel van de oorspronkelijke specificatie is verdeeld in modules en er komen ook nieuwe modules bij. Al die modules worden onafhankelijk van elkaar ontwikkeld en daardoor hebben ze ook verschillende versienummers (2, 3 maar ook 4). Vandaar dat ‘CSS versie 3’ niet bestaat; het W3C heeft het ook gewoon over CSS.

Browsers

Om webpagina's te kunnen maken en de uitleg in dit boek te kunnen volgen, hebt u geen bijzondere programma's nodig. Het maakt ook niet uit of u werkt op een pc met Windows of Linux of op een Mac met macOS (al zullen de gebruikte programma's natuurlijk iets verschillen). Elk platform gebruikt dezelfde HTML- en CSS-code.

U hebt weinig nodig om aan de slag te kunnen gaan:

- een **tekstverwerker (editor)** om HTML- en CSS-bestanden te schrijven en op te slaan;
- een **webbrowser** om te kunnen bekijken wat u maakt.

De browser is natuurlijk een belangrijk programma. Tijdens het coderen van webpagina's gebruikt u de browser om te bekijken hoe de eindgebruikers uw pagina's te zien krijgen. Gebruik bij voorkeur verschillende browsers, zodat u kunt controleren of uw pagina's ook in andere browsers goed werken. Bedenk daarbij dat in modern ontwerp verschillen in weergave tussen browsers acceptabel zijn zolang de bruikbaarheid en functionaliteit niet in het geding zijn. Verschillen ontstaan overigens zelden door de HTML-code (HTML gaat over betekenis, niet over uiterlijk), maar meestal door de CSS-code (CSS is voor de opmaak).

De meestgebruikte browsers op desktopcomputers zijn Chrome, Firefox en Edge/Internet Explorer. Op mobiel en tablets zijn dat Chrome en Safari. Hoewel het lastig is om exacte marktaandelen te achterhalen, is de grote lijn duidelijk. Op desktop heeft Chrome een aandeel van bijna 55%, Firefox 12%, Internet Explorer 12%, Edge 8% en Safari 8%. Op mobiel/tablet is de verdeling anders: Safari 46%, Chrome 37% en Samsung Internet 10%. Het aandeel van de andere browsers is op mobiel verwaarloosbaar. Dit zijn cijfers voor Nederland van 2017 en 2018 (bron: gs.statcounter.com). Europees en wereldwijd gezien zijn er wel verschillen in aandeel, maar in grote lijnen komt de verhouding overeen.



Verschillen tussen browsers

Het is nogal een open deur, maar uw website moet in elke browser goed werken. Dat wil zeggen dat alle tekst leesbaar moet zijn, links en formulieren goed werken enzovoort. Of het bedoelde ontwerp ook in elke browser tot op de pixel perfect moet zijn, hangt vooral af van uw opdrachtgever. Maar verschillen worden steeds vaker geaccepteerd, omdat er zo veel verschillende apparaten, browsers en schermafmetingen bestaan. Bovendien stellen de verschillen van vandaag weinig voor vergeleken bij de verschillen van 'vroeger', toen elke browserfabrikant zijn eigen draai aan HTML en CSS gaf.

Google Chrome

Het lijkt alsof hij er altijd al was, maar toch is Chrome, de browser van Google, pas eind 2008 gelanceerd. Tot die tijd was Internet Explorer de onbetwiste marktleider, al was het maar omdat de browser was vastgelijmd aan Windows. Chrome heeft intussen de strijd met Internet Explorer/Edge en Firefox beslist. Vanwege het grote marktaandeel mag deze browser dus niet op uw computer ontbreken. U kunt hem zelf downloaden en installeren vanaf www.google.com/chrome. Chrome volgt de webstandaarden. De browsers werd eerst gebouwd op de engine WebKit, die ook in Safari van Apple wordt gebruikt. Sinds 2013 bevat Chrome een eigen versie van WebKit genaamd Blink.



Browser engine

Een browser engine (ook wel *layout engine* of *rendering engine* genoemd) vertaalt de inhoud van een webpagina (de HTML) en informatie over opmaak (de CSS) naar een weergave op het scherm. De engine bepaalt dus welke onderdelen van HTML en CSS de browser kan gebruiken. Een CSS-eigenschap die niet in de programmacode van de engine zit, kan ook niet goed worden verwerkt. Er wordt dan gezegd dat de eigenschap niet wordt ondersteund. Bekende engines zijn WebKit (Safari), Blink (Chrome), Gecko (Firefox) en EdgeHTML (Edge).

Handig is de testversie van Chrome: Canary. Met deze ‘permanente bètaversie’ kunt u nieuwe functionaliteit testen. Canary wordt elke dag bijgewerkt en kan tegelijk met de stabiele versie van Chrome op uw computer worden uitgevoerd.



Afbeelding 1.7 Google Canary is aan de buitenkant niet anders dan Chrome, maar technisch loopt die versie altijd voor.

Chrome kan eindeloos worden uitgebreid met extensies. Kijk op de site chrome.google.com/extensions waar u in de categorie Hulpprogramma’s voor ontwikkelaars handige browseruitbreidingen vindt.

Microsoft Edge (Internet Explorer)

Internet Explorer raakte na de komst van Chrome zijn traditionele status als meestgebruikte browser al kwijt en inmiddels is ook het levenseinde in zicht, want Internet Explorer is vervangen door de nieuwe browser van Microsoft: Edge. Internet Explorer had nooit een goede reputatie onder webontwikkelaars, omdat het van alle browsers de eigenzinnigste interpretatie van webstandaarden had. Ontwikkelaars hebben daardoor jarenlang de ene hack na de andere moeten bedenken om websites compatibel te laten zijn met de verschillende versies van deze browser. In versie 11 worden de standaarden overigens wel veel beter gevuld (maar dat is dus een beetje laat).

Bij Windows 10 is de nieuwe browser Edge geïntroduceerd. Deze biedt nagenoeg dezelfde goede ondersteuning van webstandaarden als de concurrenten. Het is een browser in ontwikkeling en er zijn vergeleken met Chrome en Firefox nog niet veel extensies/add-ons voor en dat aantal groeit niet snel. Regelmatig

verschijnt een update die de functionaliteit van Edge vergroot. De thuispagina van Microsoft Edge is te vinden op www.microsoft.com/nl-nl/windows/microsoft-edge. De browser is niet als los programma beschikbaar.

Voor Edge heeft Microsoft een eigen browser engine ontwikkeld: EdgeHTML. Die heeft uiteraard geen last van het buggy verleden van Internet Explorer, maar moet wel van de grond af worden opgebouwd. Daardoor loopt Edge behoorlijk achter bij Chrome en Firefox, zeker bij de snelle vernieuwingen in CSS. Eind 2018 werd bekend dat Microsoft voor de desktopbrowser overstapt naar het open-sourceproject Chromium. Deze engine wordt onder meer in Opera gebruikt. Dit moet in de loop van 2019 merkbaar worden.

Mozilla Firefox

Korte tijd leek wereldheerschappij binnen handbereik, maar ook Mozilla Firefox is overvleugeld door Chrome. Enkele jaren geleden nog leek Firefox de leidende positie van Internet Explorer te gaan overnemen. Het zijn nu nog steeds concurrenten, maar dan voor de lagere posities. Vanwege de uitstekende implementatie van webstandaarden was het de lieveling van ontwikkelaars. Firefox is ooit ontstaan uit de failliete boedel van de oerbrowser Netscape Navigator. Programmeurs die eerst voor Netscape werkten, hebben in eigen beheer en met samenwerking van de opensourcegemeenschap een nieuwe browser ontwikkeld. Deze browser werd Firefox genoemd en in het najaar van 2004 werd versie 1.0 uitgebracht. In 2006 verscheen Firefox 2.0. Binnen enkele weken na de introductie was het programma miljoenen keren gedownload. Ook Firefox 3.0 (juni 2008) was erg populair en staat zelfs in het Guiness Book



Afbeelding 1.8 Mozilla Firefox Quantum Developer Edition is een prima browser met een goede implementatie van de webstandaarden.

of Records met het record ‘Meeste downloads in een dag’, ruim 8 miljoen. In 2018 is een nieuwe versie de browser uitgebracht: Firefox Quantum. Begin 2019 is de browser bij versie 65 en updates verschijnen met grote regelmaat, net als bij Chrome. Download Firefox vanaf www.getfirefox.com.

Net als bij Chrome zijn er veel extensies/add-ons waarmee de functionaliteit van de browser kan worden uitgebreid. Zoek bijvoorbeeld eens met de trefwoorden HTML en CSS naar extensies op addons.mozilla.org/nl/firefox.

Van Firefox is ook een versie voor ontwikkelaars beschikbaar: Firefox Quantum Developer Edition. Deze bevat nog meer nuttige hulpmiddelen voor webontwikkelaars dan de gewone editie. Bijzonder handig is de optie om het raster van CSS Grid Layout zichtbaar te maken.

MacOS: Safari

Alle hiervoor genoemde browsers zijn ook beschikbaar voor het besturingssysteem van Apple: macOS. Dat besturingssysteem heeft echter ook een eigen browser: Safari. De ondersteuning van onderdelen van HTML is vergelijkbaar met Chrome. Bij ontwikkelaars die werken op een Mac is vanwege de goede ondersteuning van standaarden en de uitstekende hulpmiddelen voor ontwikkelaars ook hier Google Chrome een populaire browser. Safari was ooit ook beschikbaar voor Windows, maar dat is niet meer zo. Het marktaandeel op de desktop is bescheiden, maar op mobiel (iPhone) en tablet (iPad) is Safari de meestgebruikte browser.

HTML-editors

Om webpagina's te kunnen schrijven, hebt u geen bijzondere of dure software nodig. Elke tekstverwerker (*editor*) waarmee tekst als kale ASCII-teks kan worden opgeslagen, is geschikt. Kladblok (voor Windows) of TekstEditor/SimpleText (voor Mac) voldoet al! Maar voor een comfortabeler leven als webdeveloper zijn er ook tal van gespecialiseerde programma's beschikbaar. Veel ervan zijn gratis.

Gespecialiseerde HTML-editors bieden allerlei voordelen: met codekleuring kunt u makkelijk de verschillende onderdelen zoals tags, attributen, eigenschappen en waarden herkennen, automatisch aanvullen van tags bespaart typewerk, spellingcontrole voor uw code scheelt debugtijd, automatisch inspringen maakt code beter leesbaar en zo is er nog veel meer.

Het aanbod van (gratis) editors is enorm en de keus is persoonlijk. Het beste advies is om er een paar uit te proberen. Verschillende websites bieden toplij-

ten aan van editors in allerlei soorten en smaken. Kijk voor een neutraal overzicht eens naar en.wikipedia.org/wiki/List_of_HTML_editors. Bekende namen zijn Visual Studio Code, Brackets (brackets.io), Atom (atom.io) en Sublime text (www.sublimetext.com). Visual Studio Code wordt hierna besproken.



Microsoft Office is geen codeertool

Gebruik niet de mogelijkheden van Word of Excel om een bestand als HTML-document op te slaan met een menuopdracht zoals Bestand, Opslaan als webpagina. Hoewel het resultaat in een moderne versie van Office niet zo rampzalig is als het ooit was, bevat de webpagina veel overbodige code. Wilt u de opmaak van een Word-document behouden, plaats dan een pdf-versie op uw website. Of maak er zelf een HTML-pagina van met CSS voor de opmaak.



CSS-editors

U hebt voor het werken met CSS geen ander programma nodig. Alle moderne editors helpen u bij het schrijven van CSS net zo goed als bij HTML, inclusief automatisch aanvullen van code, de juiste opties voor inspringen, het plaatsen van accolades en codekleuring.

Visual Studio Code

Het overweldigende aanbod van HTML-editors kan het maken een keus nogal bemoeilijken. Vindt u op het eerste gezicht niet iets van uw gading, kijk dan eens naar Visual Studio Code (VS Code), te vinden op code.visualstudio.com. De editor is gratis en beschikbaar voor Linux, macOS en Windows. Aardigheidje: de interface en alle functies zijn geschreven in HTML, CSS en JavaScript. In feite is het programma een webtoepassing die is omgezet naar een oorspronkelijk Linux-, macOS- of Windows-programma. Dat is overigens meer dan een aardigheidje. Die opbouw zorgt er namelijk voor dat de gebruiker de interface en functies eenvoudig kan aanpassen en dat het betrekkelijk eenvoudig is om er extensies voor te maken. Er zijn dan ook veel extensies beschikbaar.



Electron

Het is wel vroeg om erover te beginnen omdat u nog nauwelijks uit het startblok bent, maar u leest het goed: webtoepassingen kunnen worden omgezet naar programma's die op verschillende besturingssystemen werken. In dit geval is dat gedaan met Electron, een opensourcebibliotheek die wordt ontwikkeld door GitHub. Kijk als u er klaar voor bent eens op electronjs.org.

VS Code is een uitstekende editor voor HTML en CSS, maar ondersteunt veel meer talen, bijvoorbeeld JavaScript, TypeScript, PHP, Java en C++. De basisondersteuning van HTML en CSS is prima, met codesuggesties (IntelliSense), automatisch sluiten van tags, informatie over tags bij aanwijzen, syntaxkleuring, voorbeeldkleur, code samenvouwen en meer. Ook handig zijn de vele toetsencombinaties voor het bewerken van code: kopiëren van hele blokken, meerdere cursors tegelijk, alle keren dat een woord voorkomt ineens selecteren en meer. Op de website van VS Code staat een overzicht van toetsencombinaties. Met extensies kunt u nog veel meer handigheidjes toevoegen. De website is marketplace.visualstudio.com/VSCodium, maar u kunt net zo makkelijk zoeken vanuit de interface van VS Code.



Afbeelding 1.9 Visual Studio Code, een veelzijdige editor voor HTML en CSS.

Liveserver

In voorgaande edities van dit boek werd de editor Brackets uitgelicht. Deze editor is niet beter of slechter dan VS Code (of andere editors) en heeft zijn eigen handigheidjes. Zo is het bijvoorbeeld mogelijk om in de HTML-code met een toetsencombinatie direct de bijbehorende CSS-code te bewerken. Een ander pluspunt is de functie Live voorbeeld. Zodra u een HTML- of CSS-bestand wijzigt, ziet u direct in de browser het resultaat. Dit werkt in elk geval in Google Chrome, maar meestal ook in andere browsers. Kies daarvoor de optie Experimentele versie Live voorbeeld.

Intussen is voor VS Code een extensie beschikbaar die hetzelfde doet: Live Server. Hiermee wordt op de achtergrond een lokale webserver gestart waarin elke verandering in de code zichtbaar is zodra het document is opgeslagen. En omdat het opslaan kan worden geautomatiseerd, bijvoorbeeld direct als de cursor het codevenster verlaat, werkt dit fantastisch.



Afbeelding 1.10 Brackets.

Al die aandacht voor zoiets als een liveserver lijkt misschien overdreven. Maar zodra u intensief met het schrijven van code aan de slag gaat, zult u merken hoeveel prettiger dat werkt.

Commerciële editors/IDE's

Naast de gratis editors worden er ook editors tegen betaling aangeboden, hoewel de functionaliteit daarvan veel verder gaat dan alleen HTML- en CSS-code bewerken. Dergelijke producten vallen in de categorie geïntegreerde ontwikkelomgeving: *integrated development environment* of IDE. Ze bieden veel meer mogelijkheden dan u als beginnende ontwikkelaar nodig hebt en zelfs in de fase daarna kunt u met de genoemde gratis editors prima projecten met HTML, CSS en JavaScript uitvoeren. De meerwaarde van de commerciële producten zit vaak in de ondersteuning en nog betere en meer functies voor ontwikkeling. Voorbeelden zijn:

- [WebStorm \(\[www.jetbrains.com/webstorm\]\(http://www.jetbrains.com/webstorm\)\)](http://www.jetbrains.com/webstorm)
- [Dreamweaver \(\[www.adobe.com/nl/products/dreamweaver.html\]\(http://www.adobe.com/nl/products/dreamweaver.html\)\)](http://www.adobe.com/nl/products/dreamweaver.html)

Aanvullende hulpmiddelen

Naast browsers en een code-editor zijn er nog een paar hulpmiddelen waarmee u het schrijven en doorgronden van HTML- en CSS-code vereenvoudigt. In de voorgaande paragrafen zijn er al enkele genoemd, maar voor de duidelijkheid staan ze hier bij elkaar.

Hulpmiddelen voor validatie

Eén hulpmiddel is nog geheel niet genoemd: de validator, een hulpmiddel om te controleren of uw code aan de webstandaarden voldoet. Een validator controleert of de code voldoet aan de eisen van de taal, zoals complete tags en

verplichte attributen. U kunt een validator zien als de spellingcontrole van een tekstverwerker. U doet uw best om zo zorgvuldig mogelijk te coderen en de validator wijst u op mogelijk achtergebleven fouten.

Fouten in HTML geven lang niet altijd zichtbare problemen in de browser, maar in complexere webapplicaties kunnen ze de boel flink overhoop gooien. Met een validator bent u zulke problemen voor. Tijdens het leren van HTML is de waarde van een validator dat die laat zien waar uw code beter kan.

Op de website van het W3C zijn validators beschikbaar voor HTML en CSS, en u vindt er links naar andere validators:

- De HTML-validator staat op validator.w3.org.
- De CSS-validator is te vinden op jigsaw.w3.org/css-validator/.

De werking van de validators wijst zich vanzelf: u kunt een webadres laten controleren, een bestand uploaden of een fragment typen of plakken.

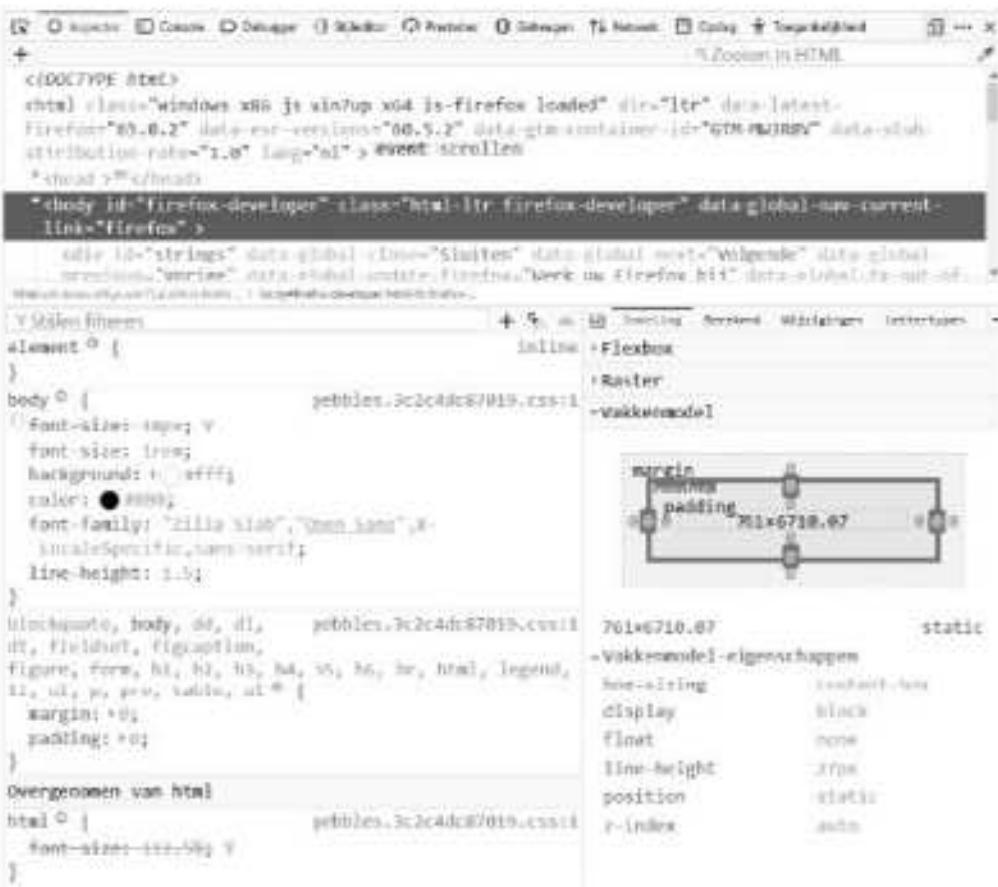


Afbeelding 1.11 HTML-validatieservice van het W3C. Voor CSS is een vergelijkbare interface beschikbaar.

Hulpmiddelen voor ontwikkeling

Er is een enorme variatie aan hulpmiddelen voor het werken met HTML- en CSS-code. Enkele voorbeelden:

- Hulpmiddelen voor ontwikkelaars in de diverse webbrowsers, eenvoudig te vinden in het menu van de browser. Deze zijn in de loop van de tijd zo uitgebreid dat u hiermee al een goed gevulde gereedschapskist hebt.
- Web Developer Toolbar voor Chrome, Firefox en Opera. Een uitbreiding die als add-on of extensie wordt toegevoegd aan de browser. Deze werk balk bevat talloze nuttige opties voor werken met CSS en HTML, formulieren, afbeeldingen enzovoort.
- De website canisuse.com met ondersteuningstabellen voor webtechnologieën (HTML, CSS, HTML API's)

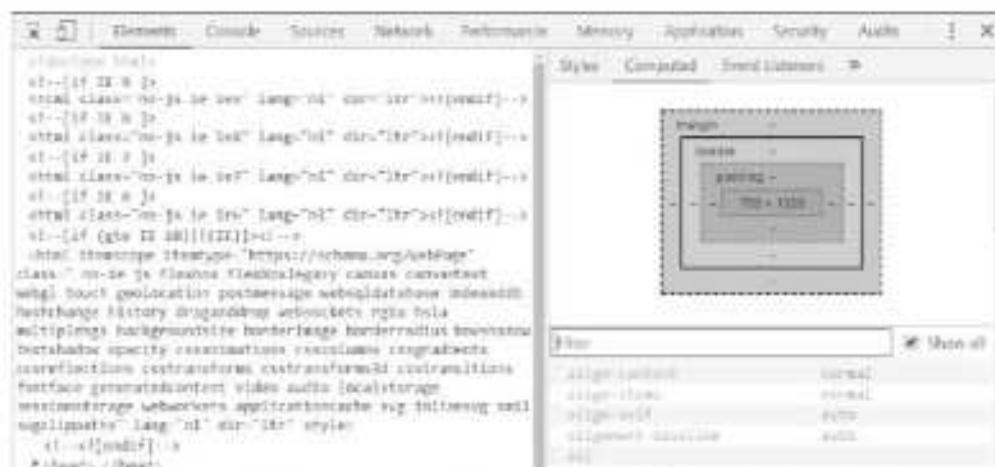


Afbeelding 1.12 Firefox Developer Edition met de Hulpprogramma's voor ontwikkelaars.

Met de hulpmiddelen voor ontwikkelaars kunt u bijvoorbeeld HTML- en CSS-code bekijken en live bewerken. Daarmee wordt experimenteren met code eenenvoudiger en leuker. Van elke webpagina, dus niet alleen die u zelf maakt, maar van elke pagina op internet, kan de code worden aangepast om te bekijken welk effect een verandering heeft (uw aanpassingen verdwijnen zodra de pagina opnieuw wordt geladen). Vooral tijdens het leerproces is het onderzoeken van door professionals gebouwde pagina's een goede tijdbesteding. Ook de ervaren webontwikkelaar profiteert van deze hulpmiddelen, omdat ze foutopsporing vereenvoudigen.

De hulpmiddelen zijn stuk voor stuk serieus gereedschap, maar het is ook gewoon leuk om ermee te spelen. De basiswerking ervan wordt grotendeels duidelijk als u ermee aan de slag gaat en voor de meer geavanceerde functies kunt u terecht op de helppagina's van de leverancier. Vanwege de grote overeenkomsten worden niet de afzonderlijke hulpmiddelen van de verschillende browsers besproken. Ter illustratie werpen we een blik op Hulpprogramma's voor ontwikkelaars van Chrome. Ze worden ook wel kortweg de Chrome DevTools genoemd. Ze zijn in het hoofdmenu van Chrome te vinden onder Meer hulpprogramma's en kunnen direct worden geactiveerd met de snel-

toets Ctrl+Shift+I. Een andere manier om ze te activeren is rechtsklikken op een onderdeel van een webpagina; kies in het snelmenu Inspecteren. Op deze manier ziet u direct alle informatie over het gekozen element.



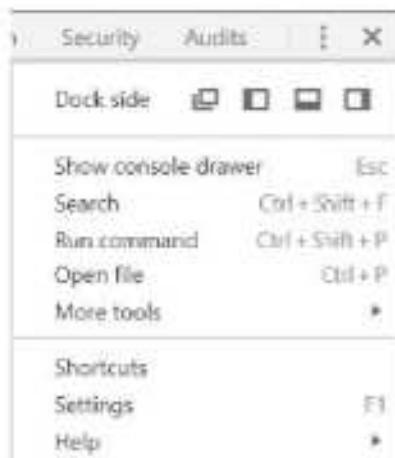
Afbeelding 1.13 Hulpprogramma's voor ontwikkelaars in Google Chrome.

Afbeelding 1.13 toont het startvenster van de DevTools. Het verschijnt standaard onder in het browservenster. Het kan ook links of rechts staan of als los venster worden getoond. De optie Dock side vindt u in het menu (afbeelding 1.14). U ziet nu direct wat in het kader van dit boek de belangrijkste informatie is: het deelvenster links toont de HTML-elementen en rechts ziet u de CSS-regels die op het geselecteerde element van toepassing zijn. Ziet u heel andere dingen, kijk dan of in de werkbalk de tab Elements actief is en in het rechterdeelvenster de tab Styles (afbeelding 1.15).

In deze weergave kunt u de opbouw van elke pagina uitzien en zien hoe elk onderdeel aan zijn opmaak komt. Op dit moment zult u nog niet alles begrijpen; we staan immers nog aan het begin. Naarmate uw kennis van HTML en CSS groeit, zult u deze informatie steeds meer waarderen.

De andere tabs (Network t/m Console) geven toegang tot functies die in dit boek niet worden behandeld. Als u eraan toe bent, vindt u alle informatie op developers.google.com/web/tools/chrome-devtools/.

Toch zijn nog twee functies ook op dit moment al de moeite van het onderzoeken waard. Links in de werkbalk (links van de tab Elements) staat een knop met een pijlpictogram. Deze doet hetzelfde als de optie Inspecteren in het snelmenu. Rechts daarvan ziet u de knop Toggle device toolbar. Hiermee schakelt u een functie in die de weergave van uw pagina op andere (mobiele) apparaten simuleert, bijvoorbeeld een Pixel 2 of iPhone X, een iPad of een laptop met HiDPI-scherm.



Afbeelding 1.15 Het menu van Google DevTools.

The screenshot shows the 'Elements' tab active. The left sidebar displays a hierarchical tree of HTML elements from the page. The root node is the entire document structure, followed by the head and body sections, then specific elements like the main header, search bar, and footer. The right panel shows the raw HTML code for the entire page, with syntax highlighting for tags and attributes.

```
<!DOCTYPE html>
<html lang="nl-NL" class="wf-roboto-i4-active wf-roboto-n3-active wf-roboto-n4-active wf-roboto-n5-active wf-robotomono-i4-active wf-roboto-i5-active wf-robotomono-n5-active wf-robotoslab-n3-active wf-robotoslab-n2-active wf-roboto-i3-active wf-robotomono-n4-active wf-active">
  <head>...</head>
  <body>...
    <a href="#main" class="visually-hidden">Direct naar de inhoud</a>
    <header>...
      <h1>...</h1>
      <div class="nav-search">
        <div>
          <input id="menu-toggle" type="checkbox">
          <label for="menu-toggle">...</label>
          <ul>...</ul>
        </div>
        ...
      </div>
    </header>
    <div class="content-wrapper">
      <main id="main">
        <article>...</article>
      </main>
      <aside class="side2">...</aside>
    </div>
    ...
  </body>
</html>
```

Afbeelding 1.14 Het tabblad Elements toont de HTML-elementen en Styles toont de CSS-opmaakregels.

- 1 Open een webpagina en activeer de DevTools (Ctrl+Shift+I).
- 2 Klik op de knop Toggle device toolbar (links van de tab Elements).

De weergave verandert en u ziet de webpagina alsof deze wordt weergeven op het geselecteerde apparaat. Als de pagina er vreemd uitziet, drukt u op F5 om hem te vernieuwen.

- 3 In de keuzelijst links van de afmetingen kiest u een ander apparaat, bijvoorbeeld Pixel 2.

Merk op dat de pagina er anders uitziet dan op uw desktop. U ziet hem zoals u hem op een echte smartphone zou zien. De standaardweergave is portret. Met de knop Rotate draait u de weergave naar landschap.

- * De cursor is veranderd in een schaduwronde dat uw vingertop represen-teert. U kunt de pagina ermee verschuiven of op een koppeling ‘tikken’. Om een element te kunnen onderzoeken klikt u op de knop Select an ele-ment... (het pijltje links van Toggle device toolbar) en klikt u op een onderdeel in de pagina. De code van het element wordt geselecteerd in het codevenster.



Geen aanwijseffect op mobiel

In DevTools ziet bij apparaten die een aanraakscherm simuleren geen aanwijseffecten, net als op uw eigen telefoon. Kiest u een gewone laptop dan ziet u ze wel. Dit is een realistische simulatie en iets om reke-ning mee te houden: laat de gebruikerservaring op aanraakschermen niet afhangen van aanwijseffecten met een muis.

Steeds vaker worden websites bekeken op mobiele apparaten. Het wordt daardoor steeds belangrijker dat websites het ook op die apparaten goed doen (of zelfs in de eerste plaats op die apparaten) en dat maakt deze optie van de DevTools bijzonder nuttig. Ontwerpen voor mobiel vraagt van de ontwikkelaar een wat andere manier van werken en stelt aanvullende eisen aan de code, maar de basis blijft hetzelfde: HTML en CSS.



Afbeelding 1.16 Simulatie van mobiele weergave.

Over dit boek

Er is voor gekozen HTML en CSS afzonderlijk te bespreken. De hoofdstukken 2 tot en met 7 gaan over HTML en de hoofdstukken 8 tot en met 13 gaan over CSS. Door deze strikte scheiding kan de uitleg worden toegespitst op het onderwerp waar het op dat moment om gaat. Het komt ook overeen met de praktijk. Een opdrachtgever biedt content aan en die moet worden gemarkerd met HTML. Aan de andere kant is er een ontwerp en dat moet worden uitgewerkt met CSS. Om de CSS te kunnen toepassen op de content, zal die eerst moeten zijn gemarkerd met HTML. Dat neemt niet weg dat u natuurlijk altijd alvast wat over CSS kunt opzoeken als u een HTML-voorbeeld wilt vormgeven. Hoe meer u zelfstandig probeert dingen voor elkaar te krijgen, hoe te meer u leert.

Oefenbestanden

Vooral bij de hoofdstukken over CSS zijn oefenbestanden beschikbaar. Vaak gaat het om de HTML- en CSS-code die is geschreven om de afbeeldingen te kunnen maken. Als de code beschikbaar is, staat de bestandsnaam in het bijschrift bij de afbeelding. Soms is er geen afbeelding en wel code. Dan staat er in de tekst een verwijzing naar het bestand.

Alle code is beschikbaar op de website bij dit boek: handboek-html-css.nl. Daar kunt u ook terecht voor op- en aanmerkingen, suggesties en vragen.

Samenvatting

Dit hoofdstuk was uw (eerste) kennismaking met HTML en CSS. Er zijn vooral algemene zaken behandeld.

- De totstandkoming van HTML 5 en welke partijen erbij zijn betrokken.
- HTML 5 omvat meer dan alleen de elementen voor het markeren van webpagina's.
- CSS wordt gebruikt voor de opmaak van HTML-documenten. Vanaf hoofdstuk 8 wordt daar dieper op ingegaan.
- Welke browsers onmisbaar en welke hulpmiddelen handig zijn.
- Een introductie van de gratis editor Visual Studio Code en de Google Hulpprogramma's voor ontwikkelaars (DevTools).

In het volgende hoofdstuk wordt dieper ingegaan op HTML: de elementen, tags en attributen. Ook wordt uitgelegd hoe met de structuurelementen van HTML 5 verschillende typen pagina's kunnen worden opgebouwd.

Oefeningen

Aan het eind van elk hoofdstuk vindt u oefeningen waarmee u het geleerde in de praktijk kunt brengen. Dit hoofdstuk bevat vooral inleidende informatie, maar toch kunt u al aan de slag.

- 1 Download en installeer verschillende browsers en zorg ervoor dat in elk geval Google Chrome of Chrome Canary is geïnstalleerd. Voor het werken met CSS Flexible Box Layout en CSS Grid Layout wordt Firefox Developer Edition van harte aanbevolen.
- 2 Verzamel informatie over HTML/CSS-editors: lees reviews en bekijk documentatie. Download en installeer een of meer editors om een indruk te krijgen van de werking.
- 3 Onderzoek de broncode van enkele van uw favoriete websites. Dat kan op verschillende manieren:
 - Open in de browser de Hulpmiddelen voor ontwikkelaars (Ctrl+Shift+I). Deze methode geeft het duidelijkste beeld van de code.
 - Alternatief: open de HTML-code in de broncodeweergave van uw browser: rechtsklik in de webpagina en kies de optie Paginabron weergeven of een vergelijkbare optie.Vergelijk de code met wat u op de pagina ziet en probeer te doorgronden wat de verschillende HTML-elementen voor functie hebben.

De basis van HTML

Na de algemene achtergronden uit het eerste hoofdstuk gaat dit hoofdstuk eerst in op de globale kenmerken van een HTML-document. Dat is onmisbare kennis als HTML nieuw voor u is. De bouwstenen van HTML worden verklaard: elementen, tags en attributen. Daarna bekijkt u wat kenmerkend is voor moderne HTML. U leert wat het contentmodel inhoudt en welke structuurelementen er zijn.

U leert in dit hoofdstuk:

De kenmerken van HTML-documenten.

Wat elementen, tags en attributen zijn.

Kenmerken van moderne HTML.

De basis van een HTML-document.

Paginastructuur en de structuurtags van HTML.

Kenmerken van HTML-documenten

Een HTML-pagina is gewoon een tekstbestand. Het bestaat uit ‘platte’ tekst. Dat wil zeggen dat er geen codes in mogen voorkomen voor paginanummering, regelafstand en dergelijke. Tekstverwerkers zoals Microsoft Word zijn daarom ongeschikt als HTML-editor.

Omdat HTML-documenten uit kale tekst bestaan, zijn ze platformonafhankelijk. Dat betekent dat u ze op ieder gewenst computersysteem of mobiel apparaat kunt maken, bekijken en bewerken.

Een webserver en een webbrowser moeten weten dat ze een HTML-bestand krijgen voorgesloten en geen gewoon tekstbestand. Daarom hebben HTML-bestanden de extensie .htm of .html. Beide zijn goed en er is technisch geen verschil.

Een kaal tekstbestand met de extensie .htm of .html wordt weergegeven door een browser, maar u ziet alleen de kale tekst. Het bestand bevat geen enkele aanwijzing voor de structuur of de betekenis van de tekst. Om aan te geven dat een tekst een kop is of een hyperlink, moet de tekst worden gemarkeerd (getagged). Daar gebruikt u de markeertaal HTML voor.

Elementen, tags en tekst

Kernbegrippen met betrekking tot HTML zijn *element*, *tag* en *tekst*. De HTML 5-standaard bevat meer dan honderd elementen. Met deze elementen beschrijft u de inhoud van een webdocument, zodat de browser weet wat die ermee moet doen. Preciezer geformuleerd: u beschrijft het document voor de *user agent*.



User agent

Een *user agent* is in dit verband software in de browser die een document dat door de webserver wordt verstuurd, vertaalt naar iets wat u kunt bekijken (of beluisteren of voelen, want er bestaan ook browsers die webpagina's voorlezen aan slechzienden – screenreaders – en er zijn browsers die webpagina's in braille weergeven). De *rendering engine* of *layout engine* verzorgt de uiteindelijke weergave in de browser (zie ook hoofdstuk 1).

Periodiek Systeem der Elementen

Afbeelding 2.1 De chemie van HTML. Ook deze website is gemaakt met HTML-elementen (bron: ptable.com).

Voor alle duidelijkheid zetten we de begrippen op een rij, daarna volgt een toelichting:

- een webpagina is opgebouwd uit elementen;
- een element kan leeg zijn, andere elementen bevatten tekst of een combinatie van andere elementen en tekst;
- het begin van een element wordt gemarkeerd door een openingstag die attributen kan hebben, soms is een attribuut verplicht;
- daarna volgt meestal tekst;
- het eind van een element wordt gemarkeerd door een sluittag;
- een leeg element is een element dat geen andere elementen kan bevatten (bijvoorbeeld het afbeeldingselement ``);
- een leeg element bestaat uit alleen de openingstag en een of meer attributen.

element	
<code> Handboek HTML5 en CSS3 </code>	
openingstag	tekst
	sluittag

Afbeelding 2.2 Een HTML-element is opgebouwd uit tags en tekst.

Opbouw van een element

Een voorbeeld maakt het allemaal duidelijker:

```
<a href="https://www.handboek-html-css.nl">Handboek HTML5 en CSS3</a>
```

Dit is een element, een hyperlink-element om precies te zijn. U ziet de volgende onderdelen:

- de openingstag ``
- het HTML-element `a`
- het attribuut `href` met de waarde `https://www.handboek-html-css.nl`
- de tekst Handboek HTML5 en CSS3
- de sluittag ``

Het hyperlinkelement is geen leeg element; het bevat tekst en heeft een sluittag. Zonder tekst zou het ook nutteloos zijn, want waar moet u dan op klikken om de hyperlink te volgen?

Een voorbeeld van een leeg element is het afbeeldingelement:

```

```

U ziet dat er alleen een openingstag is, geen tekst en geen sluittag. De tag is zelfsluitend. Er zijn wel diverse attributen, die in dit voorbeeld aangeven waar de afbeelding kan worden gevonden (`src`, de afkorting van *source* oftewel



```

```

Afbeelding 2.3 De HTML-code en het resultaat.

bron) en welke afmetingen de afbeelding heeft (height en width). Het attribuut alt beschrijft wat de afbeelding toont en is een verplicht attribuut van het element img.



Lege elementen

Andere voorbeelden van lege elementen zijn br, hr en wbr. Ze zijn bedoeld voor een regeleinde, een scheiding tussen secties en het punt waarop een woord mag worden afgebroken. U komt ze in de loop van het boek nog tegen.

Elke tag bestaat uit twee punthaken < en > met daartussen de naam van het element. Dankzij de purthaken wordt het element herkend als een tag en niet als platte tekst. Vergeet u een punthaak, dan herkent de browser de tekst niet als een tag en ziet het resultaat er op het scherm niet uit zoals u verwacht.

Een sluittag bevat altijd een slash / en de naam van het element, zoals . Een leeg element heeft geen eindtag maar is zelfsluitend. U mag een zelfsluitende tag afsluiten met een slash voor de laatste punthaak, wat er bijvoorbeeld zo uitziet: . Dit is niet verplicht en de slash 'doet' ook niets. In dit boek worden lege elementen niet afgesloten met een slash.



Element of tag?

Een element is niet hetzelfde als een tag, maar de begrippen worden vaak door elkaar gebruikt en elke webontwikkelaar begrijpt wat de bedoeling is. De tag is het ding met de punthaken, bijvoorbeeld <h1>. Tags geven het begin en eind van een element aan. Een element is het geheel van openingstag, tekst en sluittag (met uitzondering van lege elementen). Maar ook de losse naam zoals a of img wordt een element genoemd.



Speciale tag voor commentaar

Voor commentaar heeft HTML een speciale tag. Alles wat staat tussen <!-- en --> wordt genegeerd.

Elementen nesten

Hiervoor is gezegd dat elementen ook andere elementen kunnen bevatten. Een voorbeeld daarvan is:

```
<a href="https://www.mijndomein.nl">
  
</a>
```

Het element `<a>` omsluit hier het element ``. Het resultaat van deze code is een hyperlink die een afbeelding bevat. Klikt u op de afbeelding, dan komt u terecht op de bestemming die in het attribuut `href` is vastgelegd. De afbeelding is de hyperlink.

Kleine letters

Het is u misschien opgevallen dat alle voorgaande code in kleine letters is geschreven. Het hadden ook allemaal hoofdletters mogen zijn of een mix van hoofdletters en kleine letters. HTML is *case-insensitive*, hoofdletters hebben dezelfde betekenis als kleine letters. Hoewel u hoofdletters en kleine letters door elkaar mag gebruiken, is dat om uw code leesbaarder te houden niet aan te raden. Het is goed gebruik (en aanbevolen) om al uw HTML-code in kleine letters te schrijven.



Uitzondering

Het advies om altijd kleine letters te gebruiken kent één uitzondering. Het is goed gebruik om in de eerste tag in hoofdletters te schrijven: `<!DOCTYPE html>`. Het DOCTYPE wordt later uitgelegd.

Sluiten in de goede volgorde

Hier voorzag u een voorbeeld van een element dat een ander element omsluit. Het element `img` is genest in het element `a`. HTML-documenten zitten vol met dergelijke constructies. Daarbij is een ding heel belangrijk: u moet de elementen in de goede volgorde sluiten. Moeilijk is dat niet. Sluit altijd eerst het element dat als laatste is geopend. Bekijk het volgende fragment en het zal u duidelijk zijn wat de bedoeling is.

```
<p>Lees alles over <i>d</i> standaard <abbr>HTML 5</abbr></i>.</p>
```

In de volgende paragraaf zult u zien dat dit principe niet alleen geldt binnen een alinea, maar zich kan uitstrekken over vele coderegels. Sluit u een blok code niet op de goede plaats, dan kan de weergave van de pagina heel anders zijn dan bedoeld.



Ingesprongen code

Het laten inspringen van blokken code maakt beter zichtbaar of elementen goed zijn genest en of ze zijn afgesloten. Bij tekstelementen in een regel is dat een minder goed idee, maar bij alinea's en andere blokken helpt het geweldig. In afbeelding 2.5 ziet u een voorbeeld.

De HTML-code van webpagina's bekijken

Met de basiskennis uit de voorgaande paragrafen is het nuttig om eens de code van een webpagina te bekijken. Van elke webpagina die u in de browser ziet, kunt u de broncode openen. Zo is te onderzoeken hoe de ontwikkelaar de pagina heeft opgebouwd. Hier kunt u veel van leren.

- 1 Open uw browser en ga naar www.w3.org/TR/ (de pagina met alle webstandaarden voor HTML en CSS van het W3C).



Afbeelding 2.4 De pagina met alle webstandaarden voor HTML en CSS van het W3C.

- 2 Bekijk de broncode van deze pagina. Dat kan op twee manieren.

- Klik met de rechtermuisknop in de webpagina en kies Paginabron weergeven (Google Chrome). Andere browsers hebben daarvoor een vergelijkbare opdracht. De achterliggende code van de pagina wordt nu zichtbaar gemaakt in een afzonderlijk venster of een nieuw tabblad, waarbij verschillende kleuren voor de tags en teksten zijn gebruikt. Deze weergave toont de webpagina precies zoals de webserver die heeft verstuurd. Dit is niet de overzichtelijkste manier om code weer te geven.
- Druk op Ctrl+Shift+I. Hiermee worden de Hulpmiddelen voor ontwikkelaars geactiveerd. In de standaardweergave ziet u in elk geval de HTML-code en de CSS-code voor het geselecteerde element (waarschijnlijk body). Deze weergave komt grotendeels overeen met die van de paginabron. Een belangrijk verschil is dat u nu de HTML-code ziet uit het geheugen van de browser. Om precies te zijn ziet u het Document Object Model; wat dat is en wat u ermee moet wordt verderop uitgelegd in de paragraaf met dezelfde naam. Nu kunt u rechtsklikken op de tag <html>; klik in het snelmenu op Expand recursively. Na korte tijd is alle code uitgevouwen.



Chrome DevTools

Aangezien Chrome de meestgebruikte browser is, zijn de genoemde instructies en menuopties gebaseerd op de Chrome DevTools. De ontwikkelaarshulpmiddelen van andere browsers bieden grotendeels dezelfde mogelijkheden.

```
<html>
  <head>
    <link href="css/main.css" rel="stylesheet" type="text/css"/>
    <script src="js/main.js" type="text/javascript"></script>
  </head>
  <body>
    <div id="header">
      <div class="left">
        <a href="#">Logo</a>
        <div class="menu">
          <ul>
            <li>Home</li>
            <li>About</li>
            <li>Contact</li>
          </ul>
        </div>
      </div>
      <div class="right">
        <form action="https://www.google.com/search" method="get" style="margin-top: 10px;">
          <input type="text" name="q" value="Google" style="width: 200px; height: 30px; border: 1px solid #ccc; padding: 5px;"/>
          <input type="submit" value="Search" style="background-color: #0070C0; color: white; border: none; font-size: 14px; font-weight: bold; padding: 5px;"/>
        </form>
      </div>
    </div>
    <div id="content">
      <div class="left">
        
        <h1>Google</h1>
        <p>Google is een internationale zoekmachine die wordt gebruikt om informatie te vinden over een groot aantal verschillende onderwerpen. Google heeft meer dan 10 miljard zoekresultaten en is beschikbaar in vele talen. De zoekresultaten worden weergegeven in een lijst met links naar relevante websites. Google biedt ook een aantal andere diensten aan, zoals Google Maps, Google Translate en Google News. Google is een van de meest gebruikte websites ter wereld en heeft een grote invloed op het internet.</p>
      </div>
      <div class="right">
        <div id="ad">
          
        </div>
        <div id="ad2">
          
        </div>
      </div>
    </div>
  </body>
</html>
```

Afbeelding 2.5 De eerste regels van de broncode van dezelfde pagina.

Bekijk nu de broncode met de Hulpmiddelen voor ontwikkelaars.

- U ziet boven in de pagina codes zoals `<!DOCTYPE html>`, `<html>`, `<head>` en `<title>`, en een aantal andere elementen.
- Blader omlaag tot u de tag `<body>` ziet. Dit is het begin van de inhoud die u in de browser kunt zien.
- Probeer onderdelen van de code te begrijpen door te bekijken welk onderdeel op de pagina erbij hoort.
 - Plaats de ontwikkelaarshulpmiddelen links of rechts in het browservenster (de optie Dock side in de instellingen van DevTools).
 - Wanneer u in de ontwikkelaarshulpmiddelen een HTML-element aanzwijst, licht in de pagina het desbetreffende onderdeel op, dat is echt heel leerzaam.
- Gebruik de opdracht Element inspecteren (klikken met de rechtermuis-knop) om te zien hoe een element is opgebouwd.
 - In de ontwikkelaarshulpmiddelen staat links bovenaan de knop Select an element in the page to inspect it (Ctrl+Shift+C). Klik op die knop en wijs een onderdeel op de pagina aan om de bijbehorende code te zien oplichten. Klik op het onderdeel om naar de code te navigeren.

- 5 Wanneer een HTML-element is geselecteerd, wordt in het andere deelvenster in de tab Styles de bijbehorende CSS getoond; dat zijn de instructies die de opmaak bepalen. Uit de Engelse naam kunt u vaak al afleiden wat de functie van de eigenschappen is.
- 6 Let ook op het openen en sluiten van elementen zoals `<header>` en `<div>`, en merk onderaan de sluittags `</body>` en `</html>` op.

Dit document is een compleet en correct HTML 5-document. (Haal het eens door de HTML-validator op validator.w3.org om die bewering te controleren.) Het is in elk geval compleet in de zin van verplichte onderdelen zoals DOCTYPE, html, head, een titel en body.

Het volgt ook grotendeels een belangrijk principe: scheiding van inhoud en opmaak. In deze code ziet u namelijk vrijwel alleen HTML. De meeste opmaakkenmerken staan in verschillende stylesheets, die de browser laadt door de koppelingen te volgen in de link-elementen: `<link rel="stylesheet" href="xxx.css">`. Deze koppelingen naar stijlbladen staan altijd in de sectie head van het HTML-bestand. Om de inhoud van zo'n bestand te bekijken, activeert u de tab Sources en klikt u in de mappenlijst aan de linkerkant (voor deze website) op www.w3.org/2008/site/css. Hierin staan alle stijlbladen die deze site gebruikt.

Natuurlijk hoeft u de inhoud van deze bestanden nu niet te kunnen begrijpen. Maar u krijgt wel een beeld van hoe HTML- en CSS-bestanden eruit kunnen zien.

Styles	Computed	Event listeners	DOM Breakpoints	Properties	...
Header					
<code>element.style {</code>					
<code>}</code>					
<code>media="print, screen and (min-width: 481px)"</code>					
<code>header, nav, section, aside, footer, article, hgroup {</code>					
<code> display: block;</code>					
<code>}</code>					
<code>article {</code>					
<code> display: block;</code>					
<code>}</code>					
Inherited from <code>div#main_container</code>					
<code>media="print, screen and (min-width: 481px)"</code>					
<code>#main_container {</code>					
<code> margin-right: 25%;</code>					
<code> font-size: 10pt;</code>					
<code> line-height: 1.41667;</code>					
<code>}</code>					
Inherited from <code>body#main_w3c_org_just_template-default-single-single-post-</code>					
<code>media="print, screen and (min-width: 481px)"</code>					
<code>body {</code>					
<code> font-size: 10pt;</code>					
<code> background: #FFF url(../images/main/page_bg.jpg) repeat-x top</code>					
<code> left;</code>					
<code>}</code>					

Afbeelding 2.6 Een deel van de CSS-code die hoort bij de HTML-pagina in afbeelding 2.4.

Attributen

U hebt intussen al enkele attributen gezien, bijvoorbeeld href, src, width, height en alt. HTML-tags vertellen de browser wat de betekenis is van de tussenliggende tekst, maar vaak is het nuttig om de browser of de gebruiker aanvullende informatie te geven. Daarom kunt u bij veel elementen extra kenmerken aangeven. Deze extra kenmerken worden attributen genoemd.

Neem een hyperlink. Met alleen het element a weet de browser niet meer dan dat het een anker (*anchor*) moet zijn, maar met het attribuut href wordt het een hyperlink met een doel:

```
<a href="https://www.handboek-html-css.nl">
```

Een attribuut bestaat altijd uit een naam en een waarde:

- In dit geval is de *naam* het attribuut href.
- De *waarde* van het attribuut is https://www.handboek-html-css.nl.



Afbeelding 2.7 De opbouw van een attribuut.

Kenmerken van attributen

Attributen worden gebruikt om de functionaliteit van elementen te vergroten. Dit zijn de kenmerken:

- Elementen kunnen geen, een of meer attributen hebben.
- Attributen worden altijd in de openingstag opgegeven. Zodra de browser een element herkent, probeert deze ook de toegevoegde attributen te interpreteren. Zouden de attributen in de sluitcode staan, dan is het te laat; het element staat dan al op het scherm.
- Het attribuut wordt met een spatie gescheiden van de elementnaam.
- Attributen mogen in hoofdletters, kleine letters of een mix daarvan geschreven worden. Advies: kies één stijl en volg die consequent. Het is goed gebruik om attributen, net als tags, in kleine letters te schrijven.
- Een element kan meer attributen bevatten die elk een bepaald kenmerk beschrijven. De attributen worden met spaties van elkaar gescheiden.
- Tussen de naam en de waarde van het attribuut staan geen spaties. Een goede notatie is dus `href="https://www.w3.org"`.

- Fout is `href = "https://www.w3.org"`,
- De waarde van attributen mag tussen geen, dubbele of enkele aanhalingstekens worden gezet. Dit is goed: `href="https://www.w3.org"`, en dit: `href='https://www.w3.org'` en dit ook: `href=https://www.w3.org`. Aan u de keus. Het is goed gebruik om dubbele aanhalingstekens toe te passen.



Verplichte aanhalingstekens

Als de waarde van een attribuut spaties bevat, zijn de aanhalingstekens verplicht. Een voorbeeld is het opschrift (het attribuut `value`) van een verzendknop in een formulier: `value="Verzend het formulier"`.

- Sommige elementen hebben booleaanse attributen. Een booleaans attribuut is een soort aan-uitschakelaar. Als het attribuut er is, is de waarde van toepassing; is het attribuut er niet, dan geldt het ook niet. Een voorbeeld is een verplicht in te vullen veld in een formulier:

```
<input type="email" name="email" required>
```

Dit invoerveld van het type `email` moet worden ingevuld om het formulier te kunnen versturen omdat het attribuut `required` is toegevoegd. (Formulierelementen worden uitgelegd in hoofdstuk 7.)

Een booleaans attribuut mag ook worden geschreven als `required=""` of `required="required"`.

- Andere elementen zijn enumeratief, wat betekent dat ze juist wel een waarde moeten hebben: `true`, `false` of lege string (""):

```
<p contenteditable="true">..</p>
```

Globale attributen

Er zijn attributen die u bij elk element kunt gebruiken. Dit worden globale attributen genoemd. De HTML 5.2-specificatie noemt veertien globale attributen. WHATWG heeft daarbij ook attributen voor microdata opgenomen, terwijl die bij het W3C een eigen specificatie hebben (zie *Microdata* hierna). Dan zijn er nog wat attributen waar W3C en WHATWG verschillend mee omgaan. Als er goede browserondersteuning is, worden deze genoemd.

Een deel van de attributen heeft JavaScript nodig om wat nuttigs te kunnen doen en daarvoor is een veel uitgebreidere uitleg van JavaScript nodig dan in dit boek past. Voor de volledigheid zijn die attributen toch opgenomen, met links naar aanvullende informatie.

Niet-globale attributen komen in volgende hoofdstukken aan bod bij de relevante elementen.

accesskey

Het attribuut `accesskey` kan worden gebruikt om een sneltoets voor een element te definiëren, bijvoorbeeld voor een zoekvak. De waarde van het attribuut bestaat uit een hoofdlettergevoelige letter of een cijfer. Het is mogelijk verschillende toetsen toe te wijzen, gescheiden door een spatie, waarbij het tweede en volgende tekens het terugvalmechanisme zijn als de eerste toets niet beschikbaar is (het apparaat waarop de pagina wordt bekeken heeft bijvoorbeeld alleen een numeriek toetsenbord). In de standaardbrowsers werken de sneltoetsen in combinatie met de Alt-toets of Alt+Shift, dat verschilt (op de Mac is het Control). Wilt u de gebruiker sneltoetsen aanbieden, maak dan ook duidelijk hoe hij ze moet gebruiken. In het volgende voorbeeld kan de gebruiker op Alt+1 drukken om naar de homepage te gaan.

```
<a href="home.html" accesskey="1">Naar de homepage</a>
```

class

Het attribuut `class` geeft een element een classificatie. De specificatie stelt geen eisen aan de naam van een klasse, maar ontwikkelaars wordt aanbevolen een naam te kiezen die past bij de betekenis van het element, niet over de opmaak ervan. Met een klasse wordt aangegeven dat het element tot een of meer (CSS-opmaak)klassen behoort. Een element kan verschillende klassen hebben en dan komt er een spatie tussen de namen.

```
<article class="artikel hoofdartikel">
```

Meer over dit attribuut en de mogelijkheden voor klassennamen in hoofdstuk 8 over CSS.

contenteditable

Het attribuut `contenteditable` maakt een element bewerkbaar door de gebruiker. De waarde is `true` of `false` (of een lege string: `contenteditable=""`, wat hetzelfde is als `true`). Het instellen van een bewerkbaar gebied is eenvoudig genoeg zie bijvoorbeeld de uitleg op developer.mozilla.org/en-US/docs/Web/Guide/HTML/Editable_content), de mogelijkheden zijn interessant en de browserondersteuning is breed. JavaScript is nodig om de wijzigingen te bewaren of verder te verwerken.

```
<h2>Pas het verhaal aan</h2>
<p contenteditable="true">_</p>
```

data-*

Het attribuut `data-*` is een mooie toevoeging aan HTML. U kunt er namelijk uw eigen HTML-attributen mee maken en toch een geldig HTML-bestand hebben (zie ook Hulpmiddelen voor validatie, op pagina 19). Het sterretje kan worden vervangen door elke naam die u van past komt, mits die bestaat uit kleine letters. Een `data`-attribuut kan worden gebruikt voor opmaak met CSS (attributen van HTML zijn een van de manieren om elementen te selecteren voor opmaak, zie hoofdstuk 8). Het kan ook worden gebruikt om met JavaScript informatie te lezen uit het element of juist waarden te schrijven naar het element. Lees bijvoorbeeld www.sitepoint.com/how-why-use-html5-custom-data-attributes of developer.mozilla.org/en-US/docs/Learn/HTML/Howto/Use_data_attributes.

dir en lang

De taal van de webpagina of de taal van een afzonderlijk element, bijvoorbeeld een alinea, kunt u instellen met het attribuut `lang`. Van een pagina stelt u de hoofdtaal als volgt in op Nederlands: `<html lang="nl">`. Is een deel van de pagina in het Engels geschreven en moet de user agent (bijvoorbeeld een screenreader) dat weten, dan gebruikt u `<p lang="en">`. Zoekmachines kunnen dit attribuut gebruiken voor indexering van de pagina.

Het attribuut `dir` geeft de tekstrichting aan. De standaardwaarde is `ltr` (left to right). Is de hoofdtaal van de pagina Arabisch of Hebreeuws (of een andere taal die van rechts naar links wordt gelezen) dan gebruikt u `<html dir="rtl">`. Een derde mogelijke waarde is `auto`. Kijk voor meer uitleg daarover op www.w3.org/TR/html5/dom.html#the-dir-attribute.

draggable

Doe de test: dubbelklik in een webpagina op een woord of selecteer een afbeelding en sleep de selectie over de pagina. U zult zien dat slepen (meestal) mogelijk is. Hiervoor zorgt het attribuut `draggable`. Het kent drie toestanden: `wheel` met de waarde `true` (slepen), `nowheel` met de waarde `false` (niet slepen) of `auto` (geen waarde) waarbij de standaardinstelling van de browser geldt.

Slepen is leuk, maar nutteloos als u de inhoud niet ergens kwijt kunt. Daarvoor is ooit het attribuut `dropzone` bedacht, maar dat is nooit wat geworden en staat niet meer in de HTML-specificatie. Het hele slepen en neerzetten is gebaseerd op muisacties (events of gebeurtenissen) waar met JavaScript een vervolgactie aan wordt gekoppeld. Omdat het allemaal afhankelijk is van de muis, werkt dit niet met aanraakschermen. Lees er meer over in het artikel developer.mozilla.org/en-US/docs/Web/API/HTML_Drag_and_Drop_API.

hidden

Dit is een booleaans attribuut (zie ook *Kenmerken van attributen*). Als u `hidden` toevoegt aan een element geeft u daarmee aan dat de inhoud nog niet of niet meer relevant is voor de pagina. Denk aan een onlinetoets waarbij eerst een vraag moet zijn beantwoord voordat u het juiste antwoord en een toelichting te zien krijgt, of de volgende vraag. Een andere toepassing is een spelletje waarbij het spelscherm is verborgen totdat de gebruiker heeft ingelogd. Zonder JavaScript kunt u dit attribuut nauwelijks zinvol gebruiken. Het resultaat komt overeen met de CSS-regel `display: none`. In hoofdstuk 9 leert u dat dit een ander effect heeft dan de CSS-verdwijntruc met `visibility: hidden`.

```
<article hidden>
  <h2>Vraag 2</h2>
```

id

Het attribuut `id` geeft een element een unieke naam (*identifier*). Die naam mag in een webpagina daarom maar één keer voorkomen; twee elementen mogen niet dezelfde `id` hebben. De waarde van `id` moet minimaal één teken lang zijn en mag geen spaties bevatten. De waarde van `id` mag bestaan uit alleen cijfers, met een cijfer of een underscore beginnen, alleen uit leestekens bestaan enzovoort. Alleen spaties zijn niet toegestaan.

Een `id` kan bijvoorbeeld worden gebruikt als markering voor de bestemming van links binnen een pagina (anker) en als aangrijppingspunt voor scripting en opmaak met CSS. Dit attribuut komt ook aan de orde in hoofdstuk 8 over CSS.

```
<aside id="unieke_naam">
```

spellcheck

Het attribuut `spellcheck` doet wat u vermoedt. Het attribuut kan de waarden `true` (ingeschakeld) en `false` (uitgeschakeld) hebben. U kunt spellingcontrole activeren voor tekstwaarden in formulierinvoervelden (`<input>`), in het element `<textarea>` en bewerkbare tekst (elementen met het attribuut `contenteditable`). De controlectaal is de taalversie van de browser.

style

Het attribuut `style` kan worden gebruikt om een afzonderlijk element op te maken met CSS. Dat het kan, betekent niet dat het de beste manier is om elementen op te maken. Hoe meer opmaakregels er buiten stijlbladen wordt gebruikt, hoe lastiger het onderhoud van de pagina's wordt. Het kan wel handig zijn om snel wat opmaakregels te testen. U komt dit attribuut ook tegen in hoofdstuk 8 over CSS.

```
<p style="color: green; font-weight: bold;">
```

tabindex

Met `tabindex` kunt u instellen in welke volgorde elementen worden geactiveerd (de focus krijgen) wanneer de gebruiker binnen een webpagina drukt op de Tab-toets (of op Next tikt op een smartphone). Een nummer bepaalt de volgorde. Dit werkt standaard sowieso op formulierelementen en links (ook zonder het attribuut te gebruiken), maar kan op elk element worden ingesteld. De standaardvolgorde is de volgorde van de elementen in de code. In het algemeen kunt u er het best voor zorgen dat die volgorde logisch is en de volgorde niet met `tabindex` aanpassen. Situaties waarin het attribuut wel wordt gebruikt, vallen buiten het bestek van dit boek. Kijk op www.w3.org/TR/html5/editing.html#attr-tabindex voor meer details.

title

Let op: dit *attribuut title* is niet hetzelfde als het *element <title>* van de pagina. Het attribuut `title` kan een beschrijving geven van het element, bijvoorbeeld in een tooltip. Bij een link kan het de titel of een beschrijving van het doel zijn, bij een afbeelding een beschrijving van het beeld of de naam van de fotograaf, bij een alinea een noot of opmerking, enzovoort. De waarde van `title` bestaat uit tekst.

Het is niet verstandig om `title` te gebruiken voor onmisbare informatie, want zonder muis (bijvoorbeeld op mobiele apparaten) wordt de inhoud van `title` niet weergegeven. Het W3C heeft om die reden het liefst dat u dit attribuut helemaal niet gebruikt.

```

```

translate

Geen enkele browser ondersteunt (voorjaar 2019) het attribuut `translate`. Het wordt vermeld voor de volledigheid. Dit attribuut (waarden `true`, `false`, lege string) geeft aan of de teksthoud of attribuutwaarden mogen worden vertaald (bijvoorbeeld door een vertaaldienst van Google of Microsoft). Zolang dit attribuut niet wordt ondersteund, kan met de klasse `.notranslate` worden voorkomen dat Google-vertalen delen van een webpagina vertaalt.

```
<p translate="false">De quick brown fox jumps over the lazy dog</p> <-- werkt niet -->
<p class="notranslate">De quick brown fox jumps over the lazy dog</p> <-- werkt wel -->
```

In HTML 5.3 (concept)

In het concept voor HTML 5.3 (begin 2019) is de lijst globale attributen langer. Of ze allemaal de eindstreep halen is pas zeker als deze versie een aanbeveling is geworden.

De mogelijk nieuwe attributen zijn:

- **aria-*** Dit zijn attributen voor toegankelijkheid die al deel uitmaakten van de HTML-specificatie (zie de paragraaf *WAI-ARIA verderop*). Het enige verschil is dat ze onder globale attributen zijn geplaatst.
- **is** Bedoeld voor het maken van aangepaste HTML-elementen (*custom elements*). Een mooi voorbeeld (kennis van JavaScript vereist) is te vinden via developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/is.
- **nonce** Dit is een cryptografisch *number used once* en is bedoeld voor beveiliging. Ook hiervoor is kennis van JavaScript vereist. Zie bijvoorbeeld stackoverflow.com/questions/42922784/what-s-the-purpose-of-the-html-nonce-attribute-for-script-and-style-elements.
- **role** Ook dit attribuut heeft te maken met toegankelijk. De waarde geeft een welke functie (rol) het element heeft. Bij een `<article>` is dat wel duidelijk, maar bijvoorbeeld een `div` heeft geen vastomlijnde rol. Dit attribuut vertelt de browser (met name spraak of braille) welke functie het element heeft. Voorbeelden van rollen zijn `button`, `heading` en `search`.

Microdata

Microdata is een gestandaardiseerde manier om gestructureerde gegevens aan een website toe te voegen, bijvoorbeeld over een boek, film, bedrijf of recept. Denk aan zaken als titel, auteur, adres, openingstijden, ingrediënten enzovoort. Andere systemen kunnen de data gebruiken, bijvoorbeeld zoekmachines.

In HTML Living Standard zijn ook de attributen voor microdata opgenomen. Het W3C heeft daarvoor een afzonderlijke specificatie gemaakt: HTML Microdata, te vinden op www.w3.org/TR/microdata/. Meer informatie is bijvoorbeeld te vinden op schema.org/BreadcrumbList (broodkruimelspoor op de webpagina) of developers.google.com/search/docs/guides/intro-structured-data.



Rich snippets

Met microdata kunnen *rich snippets* worden gemaakt. Een rich snippet is een zoekresultaat in Google dat meer opvalt doordat er bijvoorbeeld een afbeelding bij staat of een beoordeling. Zie voor uitleg developers.google.com/search/docs/guides/mark-up-content.

WAI-ARIA

Een laatste categorie globale attributen betreft Accessible Rich Internet Applications (WAI-ARIA). Mensen met een beperking moeten net als ieder ander webinhoud kunnen raadplegen. Zij doen dat bijvoorbeeld met behulp van een braillebrowser of een spraakbrowser. De structuurtags van HTML 5 dragen al

bij aan een beter toegankelijkheid van het web. Daarnaast is er een specificatie die beschrijft hoe de betekenis van HTML-elementen verder kan worden verduidelijkt met behulp van attributen voor toegankelijkheid. Dit onderdeel heeft een eigen specificatie: www.w3.org/TR/wai-aria/.

Het onderwerp toegankelijkheid (*accessibility*) wordt door webbouwers nog vaak onderschat, met als gevolg dat mensen met een beperking lang niet alle websites goed kunnen gebruiken. In oktober 2018 bleek uit onderzoek van de Stichting Accessibility dat commerciële websites, (webwinkels, reisbureaus, energiebedrijven) gemiddeld dertien soorten obstakels hebben voor mensen met een functiebeperking.

Het W3C heeft een praktische gids voor het toegankelijk maken van website: www.w3.org/TR/2018/WD-using-aria-20180927/. Ook nuttig zijn de Richtlijnen voor toegankelijkheid van webcontent: www.w3.org/Translations/WCAG20-nl/.

Kenmerken van HTML 5

HTML 5 is niet een compleet nieuwe standaard. Het borduurt voort op HTML 4 en XHTML, en als u een webpagina bekijkt die met HTML 5 is gecodeerd, ziet die er in grote lijnen hetzelfde uit als een HTML 4-pagina. HTML 5 – en dan bedoelen we de taalspecificatie, niet alles eromheen – is vooral conceptueel anders; het is gericht op structuur en betekenis. Alles wat met opmaak heeft te maken is overboord gezet en dat heeft geleid tot het verbannen van elementen, maar ook tot het aanpassen en toevoegen ervan. Bij de ontwikkeling is goed gekeken naar hoe webbouwers in de praktijk werken. Het resultaat daarvan is onder meer dat u nu probleemloos filmpjes of muziek in uw website kunt opnemen. Maar ook minder opzichtige vernieuwingen veraangenamen het leven van de bouwers en gebruikers van websites, zoals nieuwe invoerelementen voor formulieren en nieuwe elementen voor structurering van pagina's. Dat laatste maakt bijvoorbeeld de opmaak met CSS overzichtelijker.

Globaal vallen de kernpunten van de taal HTML 5 in de volgende categorieën:

- **Betekenisvolle paginestructuur** Nieuwe elementen moedigen aan meer betekenis in de pagina aan te brengen. Waar voorheen secties in de pagina met behulp van `<div>`-elementen werden gescheiden, zijn er nu nieuwe elementen om bijvoorbeeld headers, footers, artikelen, navigatieblokken en afbeeldingen met bijschrift in op te nemen.
- **Strikte scheiding tussen inhoud en opmaak** Alle opmaakelementen en -attributen zijn uit HTML gehaald of hebben een structurele betekenis gekregen. Een aardig voorbeeld is het element `<small>`. Dit element was

bedoeld om de browser automatisch tekst in een kleiner lettertype te laten weergeven; dat is opmaak. In HTML 5 betekent `<small>` ‘de kleine lettertjes’, dus een toelichting of gebruiksvoorraarden; het element geeft nu een betekenis aan. Hoe klein die lettertjes moeten zijn, bepaalt u met CSS. Opmaakelementen zoals `` (lettertype) of `<center>` (centreren) zijn obsolet (verouderd).



Obsolet in plaats van afgekeurd

Aanvankelijk werden ‘foute’ elementen en kenmerken in de HTML 5-specificatie afgekeurd (*deprecated*), maar tegenwoordig zijn dergelijke elementen obsolet oftewel verouderd (*obsolete*). Het gebruik ervan wordt (sterk) afgeraden. Er is nog wel verschil tussen wat heet *confirming* en *non-confirming*, wat erop neerkomt dat de eerste wel en de tweede niet door de HTML-validatie komt, maar in de kern moet u obsolete kenmerken gewoon niet gebruiken. Dat zijn voornamelijk HTML 4-elementen, maar ook een enkel HTML 5-elementen dat bij nader inzien niet voldeed. Daarmee is zo’n element natuurlijk niet compleet van het toneel verdwenen. Er zijn nog miljoenen webpagina’s waarin ze voorkomen en daarom zullen browsers dergelijke elementen correct blijven verwerken.

- **Foutafhandeling** In het verleden was onvoorspelbaar hoe de browser zou omgaan met een fout in de HTML-code. In de HTML 5-standaard wordt precies aangegeven hoe de browser een fout moet afhandelen. Dat lijkt voor webontwikkelaars misschien niet zo belangrijk, maar een fout is snel gemaakt. ‘Vroeger’ zou een browser raden naar wat u bedoelde en zelf bepalen hoe bijvoorbeeld een verkeerd gecodeerde tabel werd weergegeven. Dat deden browsers op verschillende manieren, wat uiteraard niet handig is. Nu gaat elke browser op dezelfde manier met een dergelijke fout om.



Foutafhandeling en structuur

Deze uitleg gaat alleen over het zichtbare deel van de foutafhandeling. Nog belangrijker is wat u niet ziet: de boomstructuur die de browser van het document opbouwt ofwel het Document Object Model (DOM). De uitleg over DOM volgt hierna, maar in dit verband is belangrijk dat de HTML 5-standaard voorschrijft hoe het DOM moet worden opgebouwd, ook als er een fout in de code zit. Als u elementen met CSS of JavaScript wilt benaderen, is het belangrijk dat elke browser dezelfde structuur opbouwt.

- **Gemoderniseerde formulierelementen** Voor het valideren van formulieren (controleren of het correct is ingevuld) hoeft niet meer per se

JavaScript te worden gebruikt. Deze controle is ingebouwd in de formulierelementen. Daarnaast zijn er nieuwe mogelijkheden toegevoegd voor onder meer kalenders, datum- en tijdinvoer en plaatshoudertekst.

- **Multimedia** Voor het weergeven van beeld en geluid hadden browsers plug-ins nodig, Flash Player (Adobe) bijvoorbeeld of Silverlight (Microsoft). Dergelijke beschermd technologieën gaan niet samen met een open web. In HTML 5 zijn daarom elementen opgenomen waarmee direct video of audio kan worden afgespeeld.



Rol wordt steeds kleiner

Voorjaar 2019 is de rol van Flash en Silverlight verwaarloosbaar. Hete gebruik ervan is gestaag afgenaomen sinds de introductie van HTML 5.

Omringende technieken

Naast de taal HTML 5 zijn er allerlei technieken ontwikkeld en in ontwikkeling om de mogelijkheden voor het werken met webinhoud te vergroten. Ze worden wel onder HTML 5 geschaard, maar de meeste vallen buiten de HTML-specificatie.

- **Tekengebied voor bitmaps** Met het elementen <canvas> wordt een tekengebied voor bitmaps gemaakt. Met HTML kan alleen het tekengebied worden gedefinieerd. De inhoud moet met JavaScript worden gemaakt of samengesteld, waarvoor opdrachten beschikbaar zijn om onder meer lijnen en cirkels te tekenen. Mogelijke toepassingen zijn het tekenen van grafieken, het maken van een compositie met bitmapafbeeldingen, werken met SVG, video enzovoort.



Handleiding voor canvas

De handleiding op developer.mozilla.org/nl/docs/Web/API/Canvas_API/Tutorial is een aanrader voor wie zich wil verdiepen in het werken met canvas.

- **Service workers** Een netwerkverbinding kan wegvalLEN en op dat moment heeft de bezoeker van de website een probleem, dat al snel het probleem van de ontwikkelaar wordt, want: weg bezoekers. Bij de introductie van HTML 5 werd daarvoor een oplossing gepresenteerd in de vorm van offline webaplicaties, kortweg AppCache genoemd. Al vrij snel bleek het niet zo'n goede oplossing. Nu zijn er service workers, waarover informatie is te vinden op bijvoorbeeld developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API/Using_Service_Workers.

- **Gegevensopslag** U kent cookies, de kleine tekstbestandjes die op uw computer worden opgeslagen waarmee websites gegevens registreren over bijvoorbeeld uw bezoek aan de site. Cookies hebben allerlei nadelen en HTML 5 introduceert nieuwe manier om gegevens te bewaren: Web Storage. Dit is bedoeld om (tijdelijk) informatie op te slaan in de lokale opslag van de browser. Dit kan per sessie (de gegevens zijn weg als de browser wordt afgesloten) of lokaal (de gegevens blijven in opslag). Lees er meer over op www.w3.org/TR/webstorage/.
- **Geolocatie** U kent het van uw mobiel of tablet: het bepalen van uw positie en deze zichtbaar maken op een kaart of delen in een sociaal netwerk. Hier voor is de API Geolocation. Toepassing is op zichzelf niet moeilijk, maar vergt wederom kennis van JavaScript. Privacy is een belangrijk aspect: Geolocation werkt alleen als de gebruiker daar toestemming voor geeft en als de verbinding is beveiligd met HTTPS. Zie ook www.w3.org/TR/geolocation-API/ en developer.mozilla.org/en-US/docs/Web/API/Geolocation_API.



API

Voor de op JavaScript gebaseerde toepassingen in HTML 5 worden API's ontwikkeld. API staat voor Application Programming Interface. Een API is een verzameling definities (programmacode) op basis waarvan een computerprogramma kan communiceren met een ander programma. Het is de bedoeling dat browserfabrikanten zo'n API opnemen in de programmacode van de browser, waardoor webdesigners en programmeurs met behulp van JavaScript de functies uit de API kunnen gebruiken. Een overzicht is te vinden op developer.mozilla.org/en-US/docs/Web/API.

De basis van een HTML 5-document

Een minimaal HTML 5-document heeft een bijzonder eenvoudige opbouw:

```
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="utf-8">
  <title>Sjabloon voor een HTML 5-document</title>
  <link rel="stylesheet" href="css/stijlen.css">
</head>
<body>
</body>
</html>
```



Gebruik Emmet

U kunt deze code overtypen in uw editor en opslaan als basisbestand voor elke nieuw HTML-bestand dat u schrijft. Een hulpmiddel is misschien handiger. De meeste editors ondersteunen Emmet, al dan niet met een plug-in. Emmet (emmet.io) is een hulpmiddel voor ontwikkelaars waarmee met sneltoetsen codefragmenten in uw bestand worden geplaatst. In Emmet zit de complete HTML 5-sjabloon achter de sneltoets !+Tab: typ in een leeg bestand het uitroepsteken en druk op de Tab-toets. In Visual Studio code is Emmet zonder plug-in beschikbaar. Uiteraard leert u het meest door zo veel mogelijk zelf code te typen, maar Emmet kan u een hoop saaie herhalingen besparen. Bovendien moet u eerst weten hoe HTML werkt, want ander weet u niet welke opdracht u Emmet moet geven.

Hoewel niet alle onderdelen uit de sjabloon vereist zijn, is het goed gebruik ze wel op te nemen in een HTML-document. De elementen worden hierna in volgorde van opkomst besproken. U kunt dit document in de browser bekijken, maar meer dan het stof op uw beeldscherm is er niet te zien. Het zichtbare deel van een website staat in de body en die is leeg.

<!DOCTYPE html>

DOCTYPE geeft aan van welk type het document is. Dat is in het geval van HTML 5 reuze eenvoudig: het type is html. De declaratie is niet hooflettergevoelig, maar het is goed gebruik om DOCTYPE in hoofdletters en html in kleine letters te schrijven.

Het specificeren van het juiste documenttype is belangrijk. Het vertelt de browser wat voor document deze kan verwachten, zodat het volgens de standaarden kan worden weergegeven. De browser werkt in de *standards mode*.

<html lang="nl">

Dit is het echte begin van het document. Het attribuut lang is niet verplicht, maar wel onmisbaar als uw webpagina wordt voorgelezen door een screen-reader (een browser voor mensen met een visuele beperking). Ook taalafhankelijke eigenschappen (spellingcontrole, woordafbreking) werken alleen goed als de taal goed is ingesteld. De taalinstelling speelt ook een rol bij zoekmachines. <html> is de zogeheten *root* van het document.

<head>

Na de openingstag <html> volgt altijd de verplichte headsectie van het document. Hiervoor is het element <head>_</head> ontworpen. De sectie <head> is een belangrijk deel van het HTML-document. In deze sectie moet u onder meer de titel van het document opgeven. De titel is het enige zichtbare onderdeel van de head (zie hierna). De head kan zo klein zijn als in het voorbeeld, maar kan ook een waslijst aan meta-elementen en verwijzingen naar stylesheets en scriptbestanden bevatten. Deze worden niet in de pagina getoond, maar zijn wel belangrijk als achtergrondinformatie of hulpmiddel voor de werking van de pagina.

<meta charset="utf-8">

Het element <meta> kan onderdak bieden aan verschillende attributen, maar charset is het enige attribuut dat u echt moet gebruiken. Dit attribuut bepaalt uit welke tekenset de browser de letters, cijfers, leestekens en meer moet halen.

Het attribuut name kan worden gebruikt om aanvullende informatie over de pagina op te nemen, die bijvoorbeeld nuttig is voor zoekmachines. Op de volgende manier verschaft u een beschrijving en trefwoorden:

```
<meta name="description" content="Het Handboek HTML5 en CSS3 ..">  
<meta name="keywords" content="HTML,HTML5,CSS">
```

De description wordt getoond in de zoekresultaten van Google. De keywords waren ooit belangrijk voor zoekmachines, maar als gevolg van misbruik wordt er niet veel waarde meer aan gehecht. Ze helpen uw pagina niet aan een hogere positie in de zoekresultaten.



Metatags die Google begrijpt

Voor de vindbaarheid van uw website is het handig om rekening te houden met de wensen die Google heeft voor metatags. Informatie is te vinden op support.google.com/webmasters/answer/79812?hl=nl.

<title>

Binnen de <head>-codes van de pagina wordt de titel van de pagina opgegeven. Hiervoor wordt het element <title> gebruikt. Een titel is verplicht en het is verstandig een logische titel te bedenken die de inhoud van de pagina zo goed mogelijk omschrijft.

De titel komt in de titelbalk of op het tabblad van de browser te staan en wordt door veel zoekdiensten samen met de beschrijving gebruikt om pagina's te indexeren. Op Google zult u eerst de titel en direct daaronder de beschrijving van de pagina zien; dit zijn daarom belangrijke gegevens. Goed gekozen kunnen ze voor mensen de reden zijn om uw pagina te bekijken.

Samengevat:

- een titel is verplicht;
- een webdocument mag maar één titel hebben;
- deze moet altijd in de sectie `<head>...</head>` staan;
- in een titel kunnen geen andere elementen worden gebruikt.



Naamloos document

We durven het bijna niet te zeggen: upload geen webpagina's met de titel *untitled document*, *untitled page* of *naamloos document*. Deze standaardtitel die veel editors aan een nieuw document geven past u natuurlijk altijd aan. Anderen vergeten dat weleens, zult u ontdekken als u deze termen googelt.

`<link>`

Het element `<link>` is niet verplicht, maar simpelweg onmisbaar omdat u daarmee (onder meer) een koppeling maakt met het bestand met de opmaakkenmerken van het HTML-document: de CSS-stylesheet. In het basisvoorbeeld ziet het er zo uit:

```
<link rel="stylesheet" href="css/stijlen.css">
```

Hierin geeft het verplichte attribuut `rel` aan wat de relatie is met het externe bestand: een stylesheet. Het attribuut `href` geeft aan waar dat bestand staat. In dit voorbeeld is dat op de webserver in dezelfde map als waarin het huidige HTML-document staat. De mogelijkheden van dit element worden in hoofdstuk 4 uitgebreider behandeld.

`<body>`

Na de header kunt u verder alle overige tekst, afbeeldingen en codes kwijt in het middendeel, de 'body' van de webpagina. Hier geeft u de eigenlijke pagina vorm. Het is dan ook niet verwonderlijk dat dit deel omsloten wordt door tags met de naam `<body>` en `</body>`. Hierbinnen wordt de inhoud van de pagina geschreven. Hier maakt u in de rest van het boek nog uitgebreid kennis mee.

Paginastructuur

In boeken over voorgaande versies van HTML kon u op dit punt beginnen aan het vullen van de webpagina met koppen, teksten, afbeeldingen, hyperlinks enzovoort. In dit boek over HTML 5 en sowieso bij modern webdesign moet daar een stap aan voorafgaan. HTML 5 is niet voor niets voorzien van nieuwe elementen om de betekenis van inhoudsblokken aan te geven. Pas als u weet welke betekenis de inhoud heeft, weet u welke elementen u nodig hebt om de inhoud te structureren. Het begrip semantiek (*semantics*) komt u vaak tegen in relatie tot HTML 5.

Een voorbeeld: dit boek bevat hoofdstukken, paragrafen en subparagrafen. De tekst is opgedeeld in alinea's, opscrimingen en terzijdes (tip of opmerking). Met andere woorden: elk stuk tekst heeft een betekenis die van invloed is op de structuur van dit boek (de 'website'). Maak nu niet gelijk de sprong naar de opmaak van die onderdelen, want u ziet wel dat een 16-punts vette tekstregel boven een 9-punts normale tekst een kop moet zijn, maar een apparaat trekt die conclusie niet. Dat apparaat heeft een code nodig die aangeeft wat de kop is en wat de alinea. Nogmaals, het gaat hier om de *semantiek*: de betekenis van de onderdelen voor de structuur. Die moet u volkomen los zien van het uiterlijk van die onderdelen.

Document Object Model – HTML DOM

Er zijn verschillende redenen om in een HTML-document te streven naar een logische, betekenisvolle (semantische) paginastructuur en correct gebruik van elementen. De belangrijkste zijn:

- foutloze weergave in elke webbrowser;
- eenvoudiger onderhoud;
- goede indexeerbaarheid door zoekmachines;
- bruikbaarheid voor alternatieve weergaveapparaten, zoals *screenreaders*;
- eenvoudiger uitwisseling van berichten tussen websites (*syndication*).

Het eerste punt, foutloze weergave in de browser, hangt nauw samen met de manier waarop de browser ervoor zorgt dat een webpagina wordt weergegeven. Globaal gebeurt er het volgende: de browser ontvangt een stroom HTML-code van de webserver en maakt daarvan een boomstructuur. Deze structuur wordt omgezet in een grafische weergave van de tekst, afbeeldingen, formulieren enzovoort. De naam van deze structuur is Document Object Model, kortweg DOM of in dit geval HTML DOM. Het DOM is geen abstract ding; u kunt het bijvoorbeeld bekijken met de hulpmiddelen voor ontwikkelaars in de diverse

browsers. Kies bijvoorbeeld in Firefox het menu Ontwikkelaar, Inspector of rechtsklik in een webpagina geopend in Chrome en kies de optie Inspecteren.

```
<header data-matrix-category="header" style="height: 111px;">
  <section data-type="top-logo" class="bld-logo" data-matrix-category="top-logo">
    <div class="container">...</div>
  </section>
  <nav class="bld-navbar navbar navbar-static affix-top" role="navigation" data-matrix-category="header:navbar">...</nav>
</header>
```

Afbeelding 2.8 Het ziet er niet heel schokkend uit, maar dit is het HTML DOM van een pagina met uitleg over het DOM (developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction).

Elk webdocument moet uit zo'n boomstructuur bestaan. De boom heeft één beginpunt, dit is het rootelement `<html>`, waarmee elke webpagina begint. Hierbinnen staan de vertakkingen. Er is een tak (dit wordt een *node* genoemd) voor de header, een tak voor de body, binnen de body weer takken voor secties met daarin koppen, alinea's enzovoort. In moderne browsers is elke tak een object. Elk object kan met CSS worden opgemaakt en met JavaScript worden benaderd en gemanipuleerd.

U kunt zich voorstellen dat de opbouw of de opmaak van een webpagina spaak loopt als er door onjuiste codering geen logische paginastruktuur is. Browsers worden zo geprogrammeerd dat ze (bijna) altijd de pagina zullen weergeven. Maar het was in de tijd voor HTML 5 de vraag of dat gebeurde op de manier die u voor ogen had. Zelfs als de paginaweergave klopte, was het nog steeds mogelijk dat het DOM een fout bevatte die later alsnog problemen veroorzaakte, bijvoorbeeld bij de uitvoering van JavaScript of het toepassen van de CSS-opmaak. De CSS-specificatie *Selectors level 3* bevat selectoren waarmee u door het DOM kunt wandelen om precies dat element te vinden dat u wilt opmaken (meer daarover in hoofdstuk 8). Vandaar dat de foutafhandeling van HTML 5 zo'n winstpunt is.

Het contentmodel van HTML 5

Volgens de oude HTML-standaarden is een element of van blokniveau (*block level*) of van regelniveau (*inline level*). Bij weergave in een browser begint een blokelement op een nieuwe regel en kan er geen ander element naast staan. Voorbeelden zijn een kop of het alinea-element `<p>`). Een element op regelniveau heeft geen regelirde en staat per definitie in een blokelement – zoals in

`<p>Ik benadruk het belang hiervan.</p>`

In HTML 5 bestaat deze onderverdeling niet meer, omdat die gaat over presentatie en zoals herhaaldelijk is benadrukt, staat dat niet in de taakomschrijving van HTML. Daarom is in HTML 5 het begrip contentmodel geïntroduceerd. Het contentmodel beschrijft de verwachte inhoud van een element. Enkele voorbeelden:

- de inhoud van een element <h1> moet een kop zijn die slaat op de inhoud van de bijbehorende sectie (de hele pagina of een artikel);
- een element <table> moet gegevens in een raster tonen;
- een element <video> wordt gebruikt om een clip af te spelen. Het is niet bedoeld om op een pagina over beeld en geluid de paragraaf over video ermee af te bakenen...

Op deze manier zijn alle elementen ingedeeld in nul of meer categorieën die zeggen welke inhoud wordt verwacht als u dat element toepast. Er wordt niets gezegd over hoe dat element in de browser wordt getoond. U bepaalt zelf of u wilt dat een kop los staat boven de tekst of in dezelfde regel wordt weergegeven. In beide gevallen moet de kop wel logisch bij die sectie horen.



Stylesheet van de browser

Een HTML-document zonder opmaak wordt door een browser nooit plat weergegeven. Elk element krijgt een opmaak mee die is vastgelegd in de stylesheet van die browser. Vandaar dat een kop altijd een grotere letter heeft en op zijn eigen regel staat, een opsomming bullets heeft en -tekst cursief wordt weergegeven. Wat u er zelf van wilt maken bepaalt u met CSS.

De volgende uitleg bij de contentcategorieën komt uit de HTML 5-standaard. Om misverstanden als gevolg van goedbedoelde vertaling te voorkomen, zijn de Engelse termen gehandhaafd.

- **Metadata content** Content die de presentatie of het gedrag voorbereid van de rest van de content of de relatie voorbereid van het document met andere documenten of andere gerelateerde informatie overdraagt.
- **Flow content** Omvat bijna alle elementen die in de body van documenten of applicaties worden gebruikt.
- **Sectioning content** Content die het bereik definieert van headers en footers.
- **Heading content** Beschrijft de kop van een sectie (gemarkeerd door sectioning content of door de kop zelf).
- **Phrasing content** Dit is de tekst van het document en omvat elementen die tekst binnen een alinea markeren. Alinea's worden gevormd door een opeenvolging van phrasing content.

- **Embedded content** Dit is content die een andere bron in het document importeert of content in een andere webtaal (bijvoorbeeld MathML of SVG).
- **Interactive content** Content die bedoeld is voor interactie met de gebruiker.

Speciale categorieën zijn:

- **Palpable content** Vrij vertaald: 'tastbare' content. Wordt hier alleen vermeld voor de volledigheid en is in de dagelijkse praktijk niet relevant.
- **Scriptondersteuning** Elementen die zelf niet worden weergegeven, maar worden gebruikt om scripts te ondersteunen.

Bijna alle elementen vallen in de categorieën flow content of phrasing content. Bent u bekend met de oude begrippen block en inline: flow content komt globaal overeen met block level (en inline level) en phrasing content met inline level. Met dat in gedachten zijn in de volgende tabel alleen de elementen in de andere categorieën opgenomen.

Metadata	Sectioning	Heading	Embedded	Interactive
base	article	h1	audio	a ¹
link	aside	h2	canvas	audio ²
meta	nav	h3	embed	button
noscript	section	h4	iframe	details
script		h5	img	embed
style		h6	math	iframe
template			object	img ³
title			picture	input ⁴
			svg	label
			video	select
				textarea
				video ²

1 alleen in combinatie met het attribuut href

2 alleen in combinatie met het attribuut controls

3 alleen in combinatie met het attribuut usemap

4 alleen als het attribuut type niet de waarde hidden heeft

U ziet enkele embedded elementen terugkomen in de categorie interactive; daar horen ze alleen bij als er attributen zijn die de gebruiker het element laten bedienen (anders is het niet interactief).

Samengevat:

- Het contentmodel is in de plaats gekomen van de oude op presentatie gebaseerde indeling block level/inline level.
- Elementen zijn gegroepeerd op basis van hun betekenis (semantiek).
- Elementen in een categorie hebben gemeenschappelijke kenmerken voor de manier waarop ze moeten worden toegepast.

Met het contentmodel als leidraad is het uw taak om op elk stukje inhoud van een webpagina het geschiktste element toe te passen. Vaak wijst dit zich vanzelf wanneer u eenmaal bekender bent met de beschikbare elementen. Bij de elementen in de categorie sectioning content en enkele nieuwe elementen in de categorie flow content is dat soms wat lastiger.

Secties markeren

HTML 5 bevat in de categorie sectioning content vier elementen om secties te markeren: `<section>`, `<article>`, `<nav>` en `<aside>`. U markeert er delen van de pagina mee die logisch bij elkaar horen en die in een inhoudsopgave zouden kunnen staan. Ze zijn een aanvulling op de kopelementen `h1` tot `h6`. Daarnaast zijn er de flowelementen `<header>` en `<footer>` die informatie geven over de sectie waar ze bij horen en bevatten bijvoorbeeld de kop, de auteursnaam of contactgegevens. Ze maken echter geen nieuwe sectie (u zou ze niet in een inhoudsopgave zetten).

Deze nieuwe elementen (inclusief het element `<main>`, dat in het volgende hoofdstuk wordt toegelicht) zijn gebaseerd op de dagelijkse praktijk. HTML-pagina's werden altijd al opgedeeld in blokken met het element `<div>`. Dit element betekent niets, maar omsluit alleen een stuk inhoud. Door het element een kenmerk (in dit voorbeeld een `id`) te geven dat iets zegt over de inhoud, wordt – in elk geval voor u – de code beter leesbaar en kunt u de elementen eenvoudiger opmaken met CSS. In een ouderwetse webpagina vindt u bijvoorbeeld de volgende code:

Code	Functie/betekenis
<code><div id="header"></code>	de kop van de pagina, bijvoorbeeld met een titel en een logo
<code><div id="navigatie"></code>	de navigatie-elementen
<code><div id="main"></code>	de hoofdinhoud van de pagina
<code><div id="artikel"></code>	een artikel
<code><div id="kolomrechts"></code>	een zijkolom met aanvullende informatie
<code><div id="footer"></code>	de voettekst van de pagina

Deze manier van markeren is niet verboden. Er zijn zelfs veel situaties waarin een `div` de enige goede oplossing is. Maar vaak kan inhoud op een betere, betekenisvollere manier worden gemarkkeerd.

Bekijkt u een dergelijke pagina in de browser, dan herkent u onmiddellijk de header of de navigatie-elementen. Een machine ziet echter alleen een aantal `div`'s zonder verdere betekenis. Een screenreader kan daardoor zonder aanvullende informatie bijvoorbeeld niet aan een slechtziende gebruiker duidelijk maken wat de navigatie is. Dat maakt de website voor die gebruiker minder toegankelijk en daardoor minder bruikbaar.



Toegankelijkheid van websites

Webtoegankelijkheid (*accessibility*) gaat over de bruikbaarheid van websites voor mensen met een beperking. Zij kunnen websites gebruiken met hulpmiddelen zoals browsers die de website voorlezen (*screenreaders*) of in braille weergeven. Die hulpmiddelen hebben daarvoor wel betekenisvolle informatie over de inhoud nodig. Dat begint met goed gemarkkeerde inhoud. Daarnaast zijn er richtlijnen en speciale attributen vastgelegd in de standaard WAI-ARIA. Deze is te vinden op www.w3.org/TR/wai-aria/.

Gaan we terug naar het voorbeeld, dan kan dezelfde code in HTML 5 als volgt worden geschreven:

HTML 4.01	HTML 5
<div id="header">	<header>
<div id="navigatie">	<nav>
<div id="main">	<main>
<div id="artikel">	<article>
<div id="kolomrechts">	<aside>
<div id="footer">	<footer>

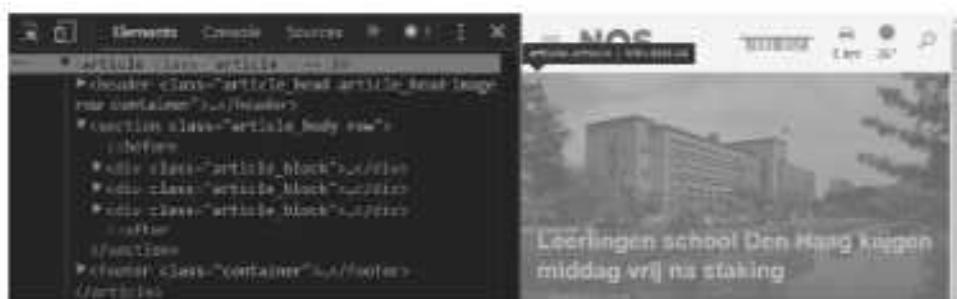
In de volgende paragrafen nemen we deze elementen door. Eerst volgt een korte uitleg van de elementen in de categorie sectioning content, aangevuld met enkele elementen uit de categorie flow content. Daarna wordt beschreven hoe u het juiste element kiest.

<article> (sectioning content)

Het element `<article>` vormt een volledig of zelfstandig geheel in een document, pagina, applicatie of site en is in principe onafhankelijk te verspreiden of te hergebruiken. Dit kan een bericht op een forum zijn, een tijdschrift- of krant-

een artikel, een blogbericht, een reactie van een lezer, een interactieve widget of elke andere zelfstandige inhoud. Een `<article>` bevat doorgaans een kop, die in een `<header>` kan zijn opgenomen met informatie over de auteur en de publicatiedatum.

`<article>` is dus bedoeld voor inhoud die compleet is en volledig op zichzelf kan staan. Als u alle inhoud van het artikel oppakt en ergens ander plaatst, buiten de oorspronkelijke context, moet het nog steeds een zinnig en compleet informatieblok zijn. Voldoet de inhoud daar niet aan, dan is het element `<article>` niet de juiste keus.



Afbeelding 2.9 Een artikel op nos.nl. Met de Huipmiddelen voor ontwikkelaars is goed te zien welke content bij welke code hoort; leerzaam!

<section> (sectioning content)

Het element `<section>` vormt een algemene sectie van een document of applicatie. Met sectie wordt in dit verband thematisch gerangschikte content bedoeld. Het onderwerp van elke `<section>` moet herkenbaar zijn, doorgaans door er een kop in op te nemen. Voorbeelden zijn hoofdstukken, pagina's in een dialoogvenster met tabs of genummerde secties in een proefschrift. Een webpagina kan in secties worden verdeeld voor een inleiding, nieuwsitems en contactinformatie.

Als de inhoud van het element geschikt is voor plaatsing op een andere website (het is een compleet geheel), dan kunt u beter het element `<article>` gebruiken. `<section>` kan ook worden gebruikt om onderdelen van een `<article>` te markeren.

`<section>` is geen algemeen containerelement. Als u een element alleen nodig hebt voor opmaak of scripting gebruikt u het element `<div>`. Een vuistregel is dat `<section>` alleen op zijn plaats is als de inhoud ervan in een inhoudsopgave zou worden opgenomen.

```

...></body>
<!---->
►<header id="nav" class="nav-wrapper">...</header>
▼<main id="content">
  ::before
  ►<section id="topstories">...</section>
  ►<div id="main">...</div>
  ►<section id="most_viewed_videos">...</section>
  ►<section id="editors_picks">...</section>
  ►<section id="nieuws_in_beeld">...</section>
  ►<section id="websites">...</section>
  ►<section id="categories">...</section>
  ::after
</main>
►<footer id="footer">...</footer>

```

Afbeelding 2.10 De homepage van nos.nl. U ziet secties voor de belangrijkste artikelen, meest bekeken video's, uitgelichte artikelen, nieuws in beeld, websites en categorieën.

```

▼<header data-metrix-category="header" style="height: 131px;">
  ►<section data-type="top-logo" class="bld-logo" data-metrix-category="top-logo">
    ►<div class="container">...</div>
  </section>
  ►<nav class="bld-navbar navbar navbar-static affix-top" role="navigation" data-metrix-category="header:navbar">...</nav>
</header>

```

Afbeelding 2.11 Hier is `<section>` niet goed toegepast. Dit zou gewoon een `<div>` moeten zijn.

<nav> (sectioning content)

Het element `<nav>` markeert een navigatiesectie: een sectie in de pagina met hyperlinks naar andere pagina's of naar delen van de huidige pagina.

U kunt de navigatie scheiden. U maakt dan een `<nav>`-sectie met links naar de pagina's in de site en een tweede `<nav>`-sectie met links naar onderwerpen op de huidige pagina. U kunt een `<nav>` een kop geven om rubrieken aan te duiden. Geef dan de tweede `<nav>` bijvoorbeeld de kop 'Op deze pagina'.

In footer's ziet u vaak hyperlinks naar de voorwaarden, de homepage en een copyrightpagina. Deze mogen wel in een `<nav>`, maar nodig is het niet. Het element `<footer>` is in dit geval voldoende.

Een `<nav>` is niet de beste oplossing voor een lijst met hyperlinks naar andere sites. Zo'n lijst is wel een kandidaat voor een ander HTML5-element: `<aside>`.

```

    <nav class="app-navigation fjs-app-navigation">
        <div class="app-navigation_sections-wrapper fjs-app-navigation-
        wrapper"> == $0
            <ul class="app-navigation_sections fjs-app-navigation-sections">
                ><li class="app-navigation_section fjs-app-navigation-section
                voorpagina app-navigation_section--active" style="border-color:
                #000000;">_</li>
                ><li class="app-navigation_section fjs-app-navigation-section
                nieuws-achtergrond" style="border-color:#008bc3;">_</li>
                ><li class="app-navigation_section fjs-app-navigation-section
                columns-opinie" style="border-color:#008bc3;">_</li>
                ><li class="app-navigation_section fjs-app-navigation-section
                video" style="border-color:#000000;">_</li>
                ><li class="app-navigation_section fjs-app-navigation-section
                wetenschap" style="border-color:#008bc3;">_</li>
                ><li class="app-navigation_section fjs-app-navigation-section
                mensen" style="border-color:#000000;">_</li>
                ><li class="app-navigation_section fjs-app-navigation-section
                de-gids" style="border-color:#000000;">_</li>
                ><li class="app-navigation_section fjs-app-navigation-section
                cultuur-media" style="border-color:#000000;">_</li>
                ><li class="app-navigation_section fjs-app-navigation-section
                foto" style="border-color:#000000;">_</li>
                ><li class="app-navigation_section fjs-app-navigation-section
                economie" style="border-color:#cf4676;">_</li>
                ><li class="app-navigation_section fjs-app-navigation-section
                sport" style="border-color:#389fef;">_</li>
            </ul>
            ><div class="app-navigation_button app-navigation_button--previous
            fjs-button-previous">_</div>
            ><div class="app-navigation_button app-navigation_button--next
            fjs-button-next">_</div>
        </div>
    </nav>

```

Afbeelding 2.12 De code voor de navigatie op volkskrant.nl.

<aside> (sectioning content)

Het element `<aside>` markeert een sectie van een pagina met daarin inhoud die 'zijdelings gerelateerd' is aan de inhoud van het element dat `<aside>` omvat en die afzonderlijk van die inhoud kan worden gelezen. Het element kan worden gebruikt voor typografische effecten zoals een citaat, voor advertenties, groepen navigatie-elementen of voor andere inhoud die los kan worden gezien van de inhoud op de pagina.

- Een `<aside>` is niet per definitie een zijkolom (*sidebar*). Dat is alleen maar de presentatie. Bij het element `<aside>` gaat het om de betekenis van de inhoud. Een `<aside>` kan er wel uitzien als een zijkolom, maar kan ook een blok midden op de pagina zijn. Een zijkolom die geen `<aside>` is, is meestal gewoon een `<div>`.
- Het omringende element bepaalt wat de inhoud van `<aside>` kan zijn. Enkele voorbeelden:

- Als u een `<aside>` opneemt in een `<article>`, kunt u daarin bijvoorbeeld een opvallend citaat uit dat artikel plaatsen. In elk geval moet de inhoud gerelateerd zijn aan dat artikel.
- Een `<aside>` direct onder `<body>` heeft betrekking op de pagina en is bijvoorbeeld geschikt voor een lijst van uw tweets, of – op een blog – koppelingen naar gerelateerde blogs.

In `<aside>` kunnen navigatiesecties worden opgenomen met `<nav>`. Daarin kan een lijst met links staan informatie die is gerelateerd aan het onderwerp waar `<aside>` bij hoort.

Voorbeeld nos.nl

Op nos.nl verschijnen begin 2019 vanwege de verkiezingen voor de provinciale staten artikelen over de Nederlandse provincies. Omdat het een reeks artikelen is, wordt bij elk artikel verwezen naar artikelen over de andere provincies. De lijst met de gerelateerde artikelen is typisch een `<aside>` bij het artikel. De `<aside>` zou ook binnen het element `<article>` moeten staan om de verbinding met het artikel duidelijk te maken.

Koppen: `<h1>` t/m `<h6>`

Een koptekst wordt in HTML gemarkkeerd met het element: `<hx>`, waarbij x een getal is van 1 t/m 6. De belangrijkste kop is `<h1>`, `<h2>` komt daaronder en `<h3>` is nog minder belangrijk. De koppen `<h4>`, `<h5>` en `<h6>` zijn van nog lager niveau; die komen in HTML-documenten trouwens zelden voor. Een kopelement heeft een openings- en een sluittag.

De kopstructuur van dit hoofdstuk zou in HTML zijn:

```
<body>
<h1>De basis van HTML</h1>
<h2>Kenmerken van HTML-documenten</h2>
<h2>Elementen, tags en tekst</h2>
<h3>Opbouw van een element</h3>
<h3>Elementen nesten</h3>
```

De eerste kop is de hoofdstuktitel en daarmee het belangrijkst. Dan volgen twee paragrafen die even belangrijk zijn. De tweede daarvan heeft twee subparagrafen.

Over koppen is veel meer te vertellen en dat gebeurt in hoofdstuk 3.

<header> (flow content)

Het element <header> bevat inleidende informatie voor de dichtstbijzijnde bovenliggende sectioning content of de *sectioning root* (waaronder het element <body>). De inhoud kan bestaan uit een kop, maar ook een inhoudsopgave, een zoekveld, een logo enzovoort.

Vertaald naar de praktijk geeft een <header> direct onder <body> informatie die van toepassing is op de hele pagina, bijvoorbeeld de paginatitel en een logo. Ook de navigatie kan erin worden opgenomen. Een voorbeeld:

```
<body>
<header>
  <h1>Vereniging van zandwientelers</h1>
  
  <nav>
    <!-- Links naar andere pagina's -->
  </nav>
</header>
```

Een <header> binnen een <article> geeft informatie over dat artikel, bijvoorbeeld de publicatiedatum:

```
<body>
<article>
  <header>
    <h1>Het moet gekker worden!</h1>
    <p>Gepubliceerd op <time datetime="2017-08-02">2 augustus 2019</time></p>
  </header>
  <p> .. </p>
</article>
```

```
*<body id="article-2276505" class="page-article js-item-read" data-
  comscore="{"name":"track.click.article"}" data-item-id="2276505">
  <!-->
  ▶<header id="nav" class="nav-wrapper">..</header>
  ▶<main id="content">
    ::before
    ▶<article class="article">
      *<header class="article_head article_head-image row container">
        ::before
        ▶<div class="article_head_meta">..</div>
        ▶<div class="article_block">..</div>
        ::after
      </header>
```

Afbeelding 2.13 Twee headers op nas.nl: een voor de pagina en een voor het artikel.



Niet overdrijven

Als een `<header>` bestaat uit een enkel kopelement, gebruik `<header>` dan niet. U bent niet verplicht elk element altijd gebruiken. De HTML-markeringen zijn bedoeld om de inhoud duidelijk te maken, niet om het webdocument als een kerstboom op te tuigen.

`<footer>` (flow content)

U kent ongetwijfeld het begrip *footer* (ook wel de voettekst genoemd). Bekijk de onderkant van wat willekeurige webpagina's en u komt de footer in alle soorten en maten tegen, van een eenvoudige copyrightvermelding tot een uitgebreid informatieblok met tekst en beeld (*fat footer*).

Het element `<footer>` markeert de footer voor de dichtstbijzijnde sectioning content of sectioning root. De footer bevat gewoonlijk informatie over de auteur, links naar gerelateerde documenten, copyrightinformatie en dergelijke. Een footer kan worden opgedeeld met `<section>` voor secties met bijvoorbeeld bijlagen, indexen, uitgebreide colofons en licentieovereenkomsten.

NOS

NOS		HOLDEDE NOS	
Meters	Over bij NOS	Wereldwijzer	Twitter YouTube
Spot	Sluiken bij de NOS	Ombudsman NPO	Facebook RSS
Uitzendhagen	Omroep	NOS-apps	Instagram
Tvnieuws	Journalistiek uitvoeringsregeling	Voorwaarden	
		Privacy	

Afbeelding 2.14 De paginafooter op nos.nl.



Contactinformatie

E-mailadressen, telefoonnummers en dergelijke informatie over auteurs of websitebeheerders horen in een element `<address>` (zie hoofdstuk 3). Dit mag op zijn beurt weer worden opgenomen in een `<footer>`.

In de praktijk zult u waarschijnlijk twee typen footers gebruiken:

- footer voor de pagina (site) met algemene informatie;
- footer bij een artikel met bijvoorbeeld de publicatiedatum en de auteur-naam.

Let op: een footer hoeft niet per se onderaan te staan; u kunt zelfs twee footers bij een artikel gebruiken – boven- en onderaan – als dat nuttig is.

Deel dit artikel  Twitter  Facebook  E-mail  WhatsApp

```
*<footer></div>
```

```
  :before
```

```
  *<div class="row">
```

```
    :before
```

```
      *<div class="col-10 col-max-8 col-centered">
```

```
        *<div class="padded-small space-bottom-xl cf">
```

```
          :before
```

```
            *<div class="share">
```

```
              <span class="share__call-to-action">
```

```
                Deel dit artikel!
```

```
              </span>
```

```
            *<ul class="share_list share_list--social">
```

```
              *<li class="share_list_item">
```

```
                *<a href="https://twitter.com/share?text=CoolCat320maakt%20familielement%20aan&url=https://nos.nl/r/2275562&ref=NOS" class="share_link js-event-click share_twitter link-grey" onclick="track_twitter_2275562()" data-comscore="{"name":"article_2275562.share","nos_platform":"twitter"}" title="deel op twitter" target="_blank">
```

```
                  *<span class="icons-wrap">
```

```
                    *<span class="icon-twitter share__icon">
```

```
                      :before
```

```
                    </span>
```

```
                  </span>
```

```
                *
```

```
                  Twitter
```

```
                *
```

```
              </a>
```

```
              *<script></script>
```

```
            </li>
```

```
          *<li class="share_list_item">...</li>
```

Afbeelding 2.15 De footer bij een artikel op nos.nl bestaat uit links naar sociale media. De afbeelding toont de HTML van één item.

Het volgende vereenvoudigde voorbeeld toont enkele mogelijkheden:

```
<body>
```

```
<article>
```

```
  <footer>Een hyperlink naar de inhoudsopgave</footer>
```

```
  ..artikeltekst
```

```
  <footer>Auteursnaam</footer>
```

```
</article>
```

```
<footer>Copyrightinformatie, voorwaarden, privacybeleid</footer>
```

```
</body>
```

Samenvatting

Dit hoofdstuk legt een belangrijke basis voor het maken van webpagina's. De theoretische achtergrond is uitgebreid, maar onmisbaar om webinhoud correct te markeren. Het volgende is aan de orde geweest:

- De kenmerken van HTML-documenten.
- Wat elementen, tags en attributen zijn en hoe ze worden gebruikt.
- Wat de toevoegingen aan HTML 5 zijn en welke oude HTML-elementen zijn vervallen.
- Welke basiselementen een HTML 5-document moet en mag bevatten: DOCTYPE, `<html>`, `<head>`, `<meta>`, `<link>`, `<title>` en `<body>`.
- Wat het HTML DOM is en waarom een correcte documentstructuur belangrijk is.
- De hoofdstructuur (zeg maar wat de inhoudsopgave zou kunnen zijn) van een webpagina wordt gemaakt door de inhoud te markeren met de elementen voor sectioning content: `<article>`, `<section>`, `<nav>` en `<aside>` en daarbij koppen van het juiste niveau te gebruiken: `<h1>` tot en met `<h6>`. De betekenis van de inhoud bepaalt welk element het best kan worden gekozen.
- Content die buiten de inhoudsopgave valt maar wel een logische eenheid vormt, groepeert u met de elementen voor flowcontent zoals `<header>` en `<footer>`. Ook hier geldt: de betekenis van de inhoud bepaalt welk element de beste keus is.

In het volgende hoofdstuk gaat het over de elementen waarmee tekst wordt gemarkkeerd: koppen en alineatekst, maar ook tekst waarop nadruk ligt of die een andere speciale betekenis heeft. Ook het maken van lijsten, het markeren van illustraties en het gebruik van bijzondere tekens (accentletters, euroteken en dergelijke) komen aan de orde.

Oefeningen

- Het is leerzaam om te bekijken hoe verschillende websites in de praktijk omgaan met de aanbevelingen voor het markeren van secties in de HTML 5-specificatie. Gebruik daarvoor de Inspector van de browser (Ctrl+Shift+I). Ga er niet van uit dat de elementen op elke site goed worden toegepast (want dat is niet zo). Kijk er kritisch naar en bedenk hoe u het zelf zou doen. U zult trouwens merken dat bij veel sites massa's div's worden gebruikt om opmaakrasters te bouwen (lees in hoofdstuk 10 hoe dat anders kan). Probeer daar doorheen te kijken. Ook wordt section vaak gebruikt om delen van de pagina af te bakenen waar de betekenisloze `<div>` een betere keus zou zijn.

- Neem een tijdschrift en zoek daar artikelen in op verschillende onderdelen, zoals koppen, citaten en kaders. Bouw deze hoofdstructuur na in een HTML-bestand met de elementen die in dit hoofdstuk zijn behandeld.
 - * Mocht u papierloos leven: neem een website als voorbeeld en bouw de hoofdstructuur na. Kijk daarna hoe de ontwikkelaar van de site het heeft aangepakt. Een goed voorbeeld is html5doctor.com.

Tekst markeren

In hoofdstuk 2 is de basis van een HTML5-document uitgelegd en is beschreven hoe u inhoud structureert en groepeert. Dit hoofdstuk gaat over de mogelijkheden om tekst te markeren, bijvoorbeeld als kop of alinea. Die twee zijn echter nog maar het begin. Er zijn elementen voor tekst met een speciale betekenis, bijvoorbeeld een boektitel of een afkorting. Ook voor gegroepeerde tekst zoals (genummerde) lijsten zijn er elementen, net als voor illustraties: grafieken of foto's. Voor speciale tekens zoals accentletters zijn er speciale codes. Daarmee kunt u elke vreemd teken in een webpagina gebruiken.

U leert in dit hoofdstuk:

Tekst invoeren: koppen en alinea's.

Elementen voor speciale betekenis van tekst, zoals nadruk of een afkorting.

Lijsten met nummers of opsommingstekens maken.

Illustraties markeren met <figure> en <figcaption>.

Speciale tekens en symbolen invoegen.

Inleiding

In hoofdstuk 2 hebt u belangrijke basiskennis opgedaan:

- de hoofdstructuur van de pagina wordt bepaald met de elementen uit de categorie sectioning content en de koppen;
- de keuze voor een element is gebaseerd op de betekenis van de inhoud en de relatie tot omringende inhoud.

Vooral het tweede punt speelt ook in dit hoofdstuk een belangrijke rol. Of u nu inhoud voor een website krijgt aangeleverd van een opdrachtgever of er zelf voor zorgt: het moet allemaal worden voorzien van tags die de betekenis van de inhoud het best weerspiegelen. Dit hoofdstuk gaat daarover: het markeren van koppen, subkoppen, alinea's, citaten en andere tekstelementen.



Ga niet af op het uiterlijk

Alle code die u schrijft en in een browser bekijkt, wordt opgemaakt met de ingebouwde stylesheet van die browser. Deze opmaak is niet meer dan een hulpmiddel bij het markeren van de inhoud. De echte opmaak wordt later toegevoegd in een stylesheet.

Koppen in HTML-documenten

Het getal bepaalt de keus, niet de tekstgrootte

Door de standaard ingebouwde opmaakkenmerken geeft de browser de koppen weer met verschillende afmetingen, waarbij `<h1>` twee keer zo groot is als standaardtekst en `<h6>` bijna half zo klein als standaardtekst. Een bezoeker die de pagina bekijkt, ziet daardoor hoe belangrijk een kop is. Maar een bezoeker die deze de pagina laat voorlezen door een spraakbrowser mist die visuele informatie en is afhankelijk van de markering.

Het is daarom nadrukkelijk niet de bedoeling dat u een kopniveau kiest dat mooi past vanwege de tekstgrootte. De enige juiste afweging bij het kiezen van een kopniveau is de plaats in de hiërarchie. Hoe die kop eruitziet regelt u met CSS.

Praktisch: als de titel van uw website de kop `<h1>` heeft, heeft een artikel een kop `<h2>` en een tussenkop in dat artikel `<h3>`. Misschien komt het formaat van een kop `<h3>` u beter uit als kop van het artikel. Dan kiest u toch `<h2>` en past u de tekstgrootte aan met CSS.

Titel website is niet automatisch <h1>

Deze eenvoudige regel betekent niet dat de keus altijd eenvoudig is. Het betekent ook niet dat de naam van uw website automatisch <h1> moet zijn. Dat kan wel, maar bijvoorbeeld op een pagina met één blogbericht is het logischer dat de titel van dat bericht <h1> is. Op een productpagina kan dat de naam van het product zijn.

Nu is het goed gebruik dat een pagina nooit meer dan één <h1>-element heeft. Maar wat moet u dan met de naam van de website/het bedrijf als <h1> al is gebruikt voor een artikel? Een mogelijkheid is om van de titel een gewone ali-nea van te maken (het element <p>, dit komt in hoofdstuk 3 aan de orde). Die kan eventueel samen met de navigatie worden opgenomen in een <header> (zie hierna). Zo is duidelijk dat het de header van de pagina is, terwijl de belangrijkste inhoud van de pagina herkenbaar is gemarkerd met <h1>. In code kan dat er zo uitzien:

```
<body>
<header>
  <p>Dit is de titel van de website</p>
  <!-- hier komen een zoekvak en een menu -->
</header>
<h1>De titel van het blogbericht</h1>
<p>Tekst</p>
<h2>Tussenkop</h2>
<p>Meer tekst</p>
```

Koppen markeren

Bij de uitleg over sectioning content in hoofdstuk 2 kwamen ze al voorbij: de kop-elementen van HTML, gemarkerd met <h1> t/m <h6>. Het belangrijkste onderwerp krijgt <h1>. Koppen die daaronder komen, bijvoorbeeld van subonderwerpen of de berichten in een blogoverzicht, krijgen kop <h2>. Een subsectie daarin krijgt kop <h3> enzovoort.

Bekijkt u dit hoofdstuk als een webpagina, dan is de titel <h1>, de kop van deze paragraaf <h2> en de paragraaf *Koppen in een header* hierna is <h3>, omdat die bij deze paragraaf hoort.

Het niveau van de kop (het nummer) wordt bepaald door te kijken naar de relatie tussen die kop en de bovenliggende inhoud. De paragraaf die verderop staat, *Tekst markeren*, hoort niet bij het onderwerp *Koppen markeren* en valt daarom direct onder de hoofdstuktitel. Vandaar: <h2>. In code:

```
<h1>Tekst markeren</h1>
<h2>Koppen markeren</h2>
  <h3>Koppen in een header</h3>
<h2>Alineateksten markeren</h2>
```

Koppen in een header

De kop van de pagina – de paginaheader – wordt vaak gecombineerd met andere elementen, zoals een subkop, een logo of een zoekvak. (Als er meer elementen zijn dan alleen de kop, worden die gegroepeerd in een `<header>`.) U hebt bijvoorbeeld de paginatitel *Mijn fantastische site* en daarbij een onderkop of tagline (slogan of slagzin): *Alle geweldige onderwerpen en meer*. U zou die subkop kunnen markeren met `<h2>`...

```
<!-- fout! -->
<header>
  <h1>Mijn fantastische site</h1>
  <h2>Alle geweldige onderwerpen en meer</h2>
</header>
```

...maar op deze manier creëert u een nieuwe sectie en dat is niet de betekenis van die onderkop. Het is een toevoeging aan `<h1>`, geen nieuw onderwerp.

Een subkop of tagline wordt daarom bijvoorbeeld als volgt gemarkkeerd:

```
<header>
  <h1>Mijn fantastische site</h1>
  <p>Alle geweldige onderwerpen en meer</p>
</header>
```

Het element `<p>` (hierna wordt het afzonderlijk besproken) geeft aan dat er een nieuwe alinea komt. Omdat die samen met de bovenliggende kop in `<header>` staat, horen ze bij elkaar.

Verdedigbaar is ook dat de subkop deel uitmaakt van de kop en daarom geen aparte alinea is. Dan kan de kop zo worden opgemaakt:

```
<header>
  <h1>Mijn fantastische site
  <span>Alle geweldige onderwerpen en meer</span>
  </h1>
</header>
```



 betekent niets

Het element `` is een betekenisloze container voor inhoud op regelniveau. Waar `<div>` blokken afbakent, markeert `` een deel alineatekst voor bijvoorbeeld een andere opmaak. Zie ook de paragraaf *Speciale betekenis aangeven*.

Bij een nieuwsbericht met een hoofdkop en een onderkop staat de onderkop soms boven de hoofdkop. Coderen kan zo:

```
<header>
  <h2>
    <span>'We vonden altijd al dat er een luchtje aan zat'</span>
    Sporen van peulvruchten ontdekt in pindakaas
  </h2>
</header>
```

Dit is nog steeds één kop, maar u kunt de elementen anders presenteren. De browser geeft deze code in één regel weer, want een `` heeft geen 'regeleinde' zoals een `<h2>` (regelniveau tegenover blokniveau). Dat wordt opgelost door met CSS van de `` ook een element op blokniveau te maken. Maar waar het hier om gaat is dat de browser 'weet' dat het citaat deel uitmaakt van de kop.

Ook voor koppen in de header van een artikel of blogbericht komt informatie aanvullend op de berichtkop u in een `<p>`-element, niet in een kop, bijvoorbeeld:

```
<header>
  <h2>Bericht uit de toekomst</h2>
  <p>Geplaatst op 1 januari 2026</p>
</header>
```

Alineatekst markeren

Tekst zou zonder markering zo in de body van het document kunnen worden gezet. Alle tekst die de browser tussen `<body>` en `</body>` tegenkomt, wordt getoond als platte tekst. Vandaar ook dat u bij een tag waar een punthaak aan ontbreekt – die daardoor niet wordt herkend als tag – wel gewoon de tekst te zien krijgt.

Op deze manier hebt u wel een paar problemen. Plakt u een in alinea's verdeelde tekst in een HTML-document, dan zijn alle regeleinden, tabs en andere

extra witruimte weg. Doe ook geen moeite om met tabs een tabel te maken in de HTML-editor, want er blijft niets van over. Het wordt een brij van tekst.

Daar komt bij dat u deze tekst niet kunt opmaken, want er is geen manier om ernaar te verwijzen in een stylesheet. Kortom: ook tekst moet worden gemarkerd.



Ingesprongen code

Met tabs of spaties in de broncode gaat de leesbaarheid ervan met sprongen vooruit en is het opsporen van fouten eenvoudiger. Plaats logische codeblokken ingesprongen ten opzichte van voorgaande code. Een goede editor doet dit (vrijwel) automatisch. Verwacht echter niet dat de browser de inspringingen overneemt.

Alinea's; het element <p>

Om in een webpagina een nieuwe alinea te beginnen, plaatst u de tekst in het element <p>. Het einde van een alinea wordt aangegeven met </p>. De letter p is de afkorting van *paragraph*, alinea in het Nederlands.

Oefening

Maak een nieuw HTML-bestand (zie ook hoofdstuk 2, *De basis van een HTML 5-document*):

```
<!DOCTYPE html>
<html lang="nl">
<head>
    <meta charset="utf-8">
    <title>Alineatekst markeren</title>
</head>
<body>

</body>
</html>
```

Typ in de body de volgende HTML-code en bekijk het resultaat in de browser:

```
<p>Om in een webpagina een nieuwe alinea te beginnen, wordt de tekst in het element &lt;p&gt; geplaatst.</p>
<p>De stylesheet van de browser voegt witruimte tussen de alinea's toe.</p>
<p>Meerdere regeleinden en returns in de broncode
```

```
worden door de browser genegeerd. Er wordt gewoon één alinea van gemaakt.</p>
```



Speciale tekens

In de broncode staat <p>. Daarmee wordt aangegeven dat letterlijk `<p>` moeten worden getoond. < en > zijn codes voor de tekens kleiner dan en groter dan: `<` en `>`. Zou u letterlijk de punthaken typen, dan ziet de browser het geheel als een tag. HTML kent veel van dergelijke codes. Ze worden besproken in de paragraaf *Speciale tekens in webpagina's*.

Het regeleinde `
`

Soms zult u in een tekst doelbewust op een volgende regel willen beginnen, maar zonder een nieuwe alinea te maken. Dit is bijvoorbeeld het geval bij adressen of gedichten. Het is in zo'n geval ook niet logisch om een nieuwe alinea te beginnen, want de tekst is een eenheid, hoort echt bij elkaar. Voor een regeleinde is er het element `
` (van *break*).



Leeg element

Merk op dat het element `
` een leeg element is. U kunt de zelfsluitende tag `
` gebruiken, maar dat is niet verplicht.

Oefening

Voeg aan het HTML-bestand dat u net hebt gemaakt de volgende code toe. Het is een gedicht (haiku). Bekijk het resultaat in de browser.

```
<p>Dikke huis, esslak,<br>
ook jij beklimt de Fuji<br>
– maar langzaam, langzaam.<br>
<code><code>Issa (1763 – 1828)</code></code>
```

Elk regeleinde is gemarkeerd met de tag `
`. Het element `<code>` markeert hier de naam van de auteur, zie de paragraaf *Special betekenis aangeven*.



Alinea's zonder witruimte

Als het u alleen te doen is om alinea's zonder witruimte ertussen (presentatie), gebruikt u niet het regeleinde `
` maar het alinea-element `<p>` met een CSS-regel die de ondermarge op 0 instelt. Marges en meer worden besproken in hoofdstuk 9.

Afbreken van woorden

Woorden in een webpagina worden in de browser niet afgebroken. Een woord dat niet op de regel past, wordt automatisch op de volgende regel gezet.

Daardoor kunnen aan het eind van de regel lelijke gaten ontstaan. U kunt dit met HTML enigszins beïnvloeden door in een erg lang woord aan te geven waar het mag worden afgebroken. Daarvoor is het element <wbr>.



Leeg element

<wbr> is een leeg element. U mag de zelfsluitende tag <wbr/> gebruiken, maar dat moet niet.

Het volgende woord kan op drie plaatsen worden afgebroken:

```
<p>Een fobie voor lange woorden heet hippo<wbr>potomonstroses<wbr>quippedalio<wbr>fobie</p>
```



CSS-eigenschap hyphens is handiger

Het element <wbr> geeft volledige controle over het afbreekpunt. De keerzijde is dat aan lange woorden van tevoren of achteraf op de gewenste plaatsen <wbr> moet worden toegevoegd, wat nogal een klus kan zijn. Bovendien ontbreekt een afbreekstreepje. De CSS-eigenschap `hyphens` lost beide problemen op. Tekst waarop deze eigenschap is toegepast, wordt automatisch afgebroken af volgens de regels van de geldende taal (bij voorkeur ingesteld met <html lang="nl">) en er wordt een afbreekstreepje toegevoegd. Helaas werkt dit met name in Chrome nog niet vlekkeloos (zie caniuse.com). Zie ook hoofdstuk 11.

Speciale betekenis aangeven

HTML bevat een aantal elementen om een speciale betekenis van tekst te markeren, bijvoorbeeld nadruk, extra belang, nadere uitleg, een citaat of een boektitel. Deze elementen vallen in de categorie *text-level semantics* (betekenis op tektniveau). Het resultaat kan zichtbaar zijn in de opmaak, maar dat hoeft niet (hoewel het voor de hand ligt). In een screenreader wordt dergelijke tekst bijvoorbeeld luider of met nadruk voorgelezen. Of en welke opmaak wordt toegekend bepaalt u met CSS, tenzij u het aan de browser zelf wilt overlaten, die de tekst bijvoorbeeld vet of cursief maakt op basis van de interne stylesheet. De kern van de zaak: de elementen worden gebruikt om de afwijkende betekenis ten opzichte van de omringende tekst aan te duiden. Ze moeten niet worden gebruikt als u alleen een andere opmaak wilt. (Daarvoor markeert u de tekst met .)

Klemtoon leggen met ``

Bedoeld om de inhoud te beklemtonen. Wordt meestal cursief weergegeven. Het gaat om klemtoon die van invloed is op de betekenis van de zin.

Als de kat het heeft gedaan en niet de hond:

```
<p>De <em>kat</em> krabt de krullen van de trap.</p>
```

Als de trap het slachtoffer is en nu eens niet uw hoofd:

```
<p>De kat krabt de krullen van de <em>trap</em>.</p>
```

Nadruk leggen met ``

Hiermee wordt benadrukt dat de inhoud belangrijk, ernstig of dringend is. Wordt meestal vet weergegeven. Screenreaders lezen met `` gemarkeerde tekst met meer nadruk voor.

```
<h1>Hoofdstuk 3: <strong>Aan de slag met HTML5</strong></h1>
```

```
<p><strong>Let op.</strong> Vergeet niet het element af te sluiten.</p>
```

```
<p><strong>Waarschuwing!</strong> Deze route kan gevaarlijk zijn. <strong>Houd rechts!</strong></p>
```

Aanvullende informatie is `<small>`

Met `<small>` markeert u aanvullende informatie; de spreekwoordelijke kleine lettertjes.

```
<p><small>Op al onze leveringen zijn de algemene voorwaarden van toepassing.</small></p>
```

```
<p>Deze bezemkast kost 350.000 euro <small>exclusief overdrachtsbelasting</small></p>
```

Ongeldig maken met `<s>`

Het element `<s>` markeert informatie die is niet meer klopt of niet meer geldig is. Wordt meestal doorgehaald weergegeven.

```
<p>Browsers te koop.<s>Nu slechts 5 euro,</s> Gratis af te halen.</p>
```

Verwijzing naar boek of film: `<cite>` en `<q>`

`<cite>` markeert een verwijzing naar een creatief werk, zoals een boek, film, artikel, liedje, website enzovoort. Het kan gaan om de titel, de auteur of een link.

```
<p>We hebben de film <cite>Captain Marvel</cite> gezien.</p>
```

<cite> is niet bedoeld voor citaten. Een citaat uit een interview, een film, een liedje enzovoort wordt gemarkkeerd met <q>. Plaats het citaat niet tussen aanhalingsstekens; dat doet de browser (of u schrijft uw eigen opmaak in CSS). Een verwijzing naar een webadres van de bron kan in het attribuut cite worden opgenomen.

```
<p>Hier staat: <q cite="https://www.trainjedraak.nl">Alleen in de openlucht proberen.</q></p>
```

Het volgende is ook mogelijk:

```
<p>In de folder <cite>Ideeën voor kinderfeestjes</cite> staat:
```

```
<q cite="http://www.trainjedraak.nl">Alleen in de openlucht proberen.</q></p>
```

Afkortingen en definities: <abbr> en <dfn>

Een afkorting kunt u markeren met <abbr>. U kunt het element gebruiken met het attribuut title om een verklarende tekst op te nemen, maar dat is niet verplicht. Het element kan ook zonder title worden toegepast als het enige doel is om afkortingen met CSS bijvoorbeeld de opmaak kleinkapitaal te geven.

```
<p>We hebben het over <abbr title="Hypertext Markup Language">HTML</abbr>.</p>
```

In dit voorbeeld zou <dfn> ook kunnen. Dit element markeert de eerste keer dat een afkorting voorkomt.

```
<p>We hebben het over <dfn><abbr title="Hypertext Markup Language">HTML</abbr></dfn>.</p>
```

Hetzelfde voorbeeld kan ook als volgt worden gemarkkeerd:

```
<p>We hebben het over <dfn title="Hypertext Markup Language">HTML</dfn>.</p>
```

Tot slot kan de afkorting met verklaring zo worden gemarkkeerd dat u ernaar kunt terugverwijzen (met een hyperlink naar het attribuut id van <dfn>):

```
<p>We hebben het over <dfn id="html"><abbr title="Hypertext Markup Language">HTML</abbr></dfn>.</p>
```

```
<p>Terugkomend op <a href="#html"><abbr title="Hypertext Markup Language">HTML</abbr></a>, ...</p>
```

Wordt in dit laatste voorbeeld geklikt op HTML, dan verspringt de pagina naar het punt waar de afkorting wordt verklaard.

Programmacode en dergelijke: <code>, <samp> en <kbd>

Er zijn enkele elementen waarmee u (programma)code, computeruitvoer en gebruikersinvoer in een HTML-pagina kunt markeren.

Met <code> kan een fragment of sleutelwoord worden gemarkleerd:

<p>Wijs de waarde toe aan a: <code>var a = 10;</code></p>

Als tabs (witruimte) in de code moeten worden behouden, combineert u <code> met <pre> (zie ook hierna):

```
<pre><code>start() {
    go
    end
}</code></pre>
```

<samp> is bedoeld voor computeruitvoer:

<p>U ziet de melding <samp>Klik op start om te stoppen</samp>.</p>

<kbd> is bedoeld voor gebruikersinvoer:

<p>Kies <kbd>Bewerken, Plakken</kdb>.</p>

Subscript en superscript: <sub> en <sup>

Voor constructies zoals H₂O en tien tot de vierde macht (10⁴) gebruikt u subscript en superscript, en de bijbehorende elementen zijn <sub> en <sup>.

<p>Jij verdient dus geld als H₂O</p>

<p>De uitkomst van 10⁴ is 10000</p>

Tekst markeren met <mark>

Het element <mark> markeert tekst. Het is niet bedoeld als markeerstift voor algemeen gebruik, maar om tekst te benadrukken die specifiek relevant is voor de lezer of pagina. Denk aan zoekmachines die de zoekterm markeren in de resultaten. Een ander voorbeeld is een artikel over een financieel gierend uit de kluw gelopen project, waarin u de projectleider citeert die een maand eerder beweerde dat alles op rolletjes liep.

Het containerelement

Het element betekent niets. Het is een container voor inhoud op regel-niveau, zodat daarop globale attributen kunnen worden toegepast, bijvoor-beeld class voor opmaak met CSS. De functie is vergelijkbaar met die van div, met als verschil dat div voor blokniveau is bedoeld.

```
<p>Een RGB-kleur bestaat uit <span class="rood">rood</span>, <span class="groen">groen</span> en <span class="blauw">blauw</span>.</p>
```

Toegevoegd of verwijderd: <ins> en

Met <ins> en geeft u aan dat inhoud is toegevoegd (*insert*) of verwij-
derd (*delete*). Het gaat hierbij om correcties op een bestaande tekst. U kunt
denken aan een wijziging van een nieuwsbericht als gevolg van nieuwe ontwik-
kelingen of verbetering van een uitleg. De correctie moet relevant zijn voor de
lezer. Als u typefouten in een bericht corrigeert, hoeft u die niet allemaal te
markeren (of u moet het zo bont hebben gemaakt dat de betekenis van het
bericht er anders door wordt). Een voorbeeld:

```
<h2>Een lay-out maken</h2>  
<p><del>Tabellen zijn het ideale hulpmiddel om een webpagina vorm te geven.</del></p>  
<p><ins>De lay-out van een webpagina bepaalt u met CSS.</ins></p>
```

Met de attributen cite en datetime kunt u aanvullende informatie geven over
de wijziging, bijvoorbeeld een link naar een bron of de datum waarop de wijzi-
ging heeft plaatsgevonden.

Inhoud groeperen

Het element <p> groepeert woorden en zinnen in een alinea. Zo zijn er meer
elementen in wat wordt genoemd de categorie *grouping content*. Het groepe-
ren moet een semantische reden hebben; de betekenis van de inhoud moet
het logisch maken dat het geheel bij elkaar wordt gehouden. Gaat het puur om
ondersteuning van de presentatie (of scripting), dan gebruikt u <div>.

<address>

Het element <address> markeert contactinformatie over het document of
een deel van het document. Het gaat om contactinformatie van de persoon
die u kunt aanspreken op de inhoud waar het element bij hoort; de beheerder
van de content. De informatie kan in een paginafooter staan (als kind van het
element <body>) en dan is er één contactpersoon voor de hele pagina.
<address> kan ook bij een artikel staan en dan is die persoon het aanspreek-

punt voor dat artikel. Bij een ander artikel kan een adreselement met andere contactinformatie staan.

De contactinformatie kan bestaan uit een naam, een e-mailadres, een postadres, een telefoonnummer enzovoort.

Merk op dat op een bedrijfssite de contactinformatie van het bedrijf niet automatisch in `<address>` thuishoort. Dat is alleen zo als op dat adres ook rechtstreeks contact kan worden opgenomen met de beheerder van de pagina. Andere contactgegevens zoals de afdeling verkoop, de directie of klantenservice hoort niet thuis in `<address>`. U kunt deze gegevens opnemen in een alinea (`<p>`) en de adrescomponenten scheiden met een `
`.

<main>

Het element `<main>` markeert de belangrijkste inhoud van een document of applicatie. De inhoud van dit element is direct gerelateerd aan – of een uitbreiding van – het hoofdonderwerp van een document of de hoofdfunctie van een applicatie. De header, footer en navigatie van de pagina horen niet thuis in `<main>`.

In een document mag `<main>` meerdere keren voorkomen, maar er mag maar één `<main>` tegelijk zichtbaar zijn. Het moet niet worden gebruikt als onderdeel van een `<article>`, `<aside>`, `<footer>`, `<header>` of `<nav>`. Dus niet zo iets:

```
<!-- fout! -->
<article>
  <h2> kop </h2>
  <p class="intro"> introtekst </p>
  <main> hoofdtekst </main>
</article>
```

Het volgende is wel goed:

```
<!-- goed -->
<main>
  <article>
    <h2> kop </h2>
    <p class="intro"> introtekst </p>
    <p> artikeltekst </p>
  </article>
  ...
</main>
```

De achtergrond van `<main>` is toegankelijkheid (accessibility). Voor een toegankelijke website werd hoofdinhoudbal vaak gemarkerd met `<div id="content" role="main">`. Hierbij is `role="main"` een speciaal attribuut dat bijvoorbeeld screenreaders vertelt waar de hoofdinhoudbegint, zodat de gebruiker daar snel naartoe kan navigeren. Het element `<main>` neemt de functie van dat attribuut over.



Gebruik een skip-link

Er handig voor mensen die navigeren met het toetsenbord is een (visueel verborgen) koppeling boven in het document die direct naar de hoofdinhoudbaat. Dit wordt wel een skip-link genoemd: daarmee slaat de gebruiker een deel van de inhoud over, bijvoorbeeld het menu. Hoe een skip-link werkt wordt uitgelegd in hoofdstuk 4.

Los van de toegankelijkheid wordt hoofdinhoudb sowieso vaak verpakt in een constructie als `<div id="content">`, omdat daarmee de hoofdinhoudb makkelijk is te benaderen met CSS en JavaScript. Deze constructie kan worden vervangen door `<main>`, al dan niet voorzien van een attribuut `id` of `class`.

Vooraf opgemaakte inhoud: `<pre>`

Het element `<pre>` hebt u hiervoor in actie gezien bij het element `<code>`. `<pre>` heeft tot doel de witruimte in opgemaakte tekst te behouden. Inspiringe code is een voor de hand liggende toepassing. De gebruikelijke volgorde van de elementen is eerst `<pre>` en dan `<code>`.

Een ander voorbeeld is ASCII-kunst (*ASCII-art*), waar uiteraard niets van overblijft (de betekenis verdwijnt) als al het wit zou worden gestript.

The ASCII art depicts a person's face and upper body. The person has short hair, a prominent nose, and is wearing a dark shirt. The style is minimalist, using only black characters on a white background.

Afbeelding 3.1 Op internet is veel ASCII-kunst te vinden.

Citaten gebruiken: <blockquote>

<blockquote> is bestemd voor inhoud die wordt geciteerd uit een andere bron. De bron van het citaat kan worden aangegeven met <cite>, al dan niet in een <footer> (van het citaat, niet van de pagina). Dit gebruik van <blockquote> is een verandering ten opzichte van eerdere versies van de HTML5-specificatie, maar omdat op deze wijze citeren algemeen gebruik was en is, is die praktijk overgenomen.

Een uitspraak uit een onlinebron kan op de volgende manier worden geciteerd:

```
<blockquote cite="https://nl.wikipedia.org/wiki/Asterix">
<p>Rare jongens, die Roneinen.</p>
<cite>Obelix</cite>
</blockquote>
```

Bij de HTML5-aanbeveling op www.w3.org/TR/2018/WD-html53-20181018/grouping-content.html#the-blockquote-element zijn diverse – uitgebreide – voorbeelden te vinden. Zeker het voorbeeld van een blog kan door de vele elementen verwarrend zijn. Houd voor ogen dat de kern eenvoudig is: plaats het citaat in <blockquote> en markeer daarbij de bron met <cite>.

Toepassing van <blockquote>, <q> en <cite>

Het kan verwarrend zijn, drie elementen voor citaten/verwijzingen. Samengevat zit het als volgt:

- <blockquote> is voor langere zelfstandige citaten uit een andere bron, waarbij de bron wordt vermeld in het attribuut cite. Een leesbare bronvermelding kan onder het citaat worden geplaatst in het element <cite>.
- <q> is voor korte citaten in een lopende zin. Ook nu kan de bron worden vermeld in het attribuut cite.
- <cite> (het element) is voor het markeren van de naam van een auteur of van een creatief werk. Het heeft geen attributen voor bronnen.

Lijsten maken: , en

Niets zo fijn als lijstjes: 10 topledjes voor in de file (lange versies), 12 beste boeken voor bij de open haard, 7 guilty pleasures in willekeurige volgorde, eenvoudig stappenplan voor een briljante website enzovoort.

Lijsten worden in HTML gemarkeerd met drie elementen:

- markeert een lijst met een rangschikking (*ordered list*);
- markeert een lijst met een willekeurige volgorde (*unordered list*);
- markeert een item in een lijst (*list item*). Dit element kan alleen worden gebruikt binnen of .

Welk type lijst van toepassing is, wordt bepaald door de inhoud van de lijst. Een rangorde, een stappenplan en andere lijsten waarbij de betekenis verandert wanneer de volgorde wordt aangepast, zijn geordende lijsten. Is de volgorde min of meer willekeurig, dan is het een niet-geordende lijst. Ongeacht het lijstype worden items gemarkerd met .

Geordende lijsten:

Een lijst waarvan de volgorde van de items belangrijk is, wordt gemarkerd met . Dit element heeft drie attributen:

- **reversed**: dit booleaanse attribuut geeft aan dat de lijst in aflopende volgorde staat.
- **start**: dit attribuut geeft het beginpunt van de lijst aan en moet een geheel getal als waarde hebben.
- **type**: dit attribuut bepaalt waarmee de items worden gemarkerd. Mogelijke waarden zijn:
 - 1 nummerde lijst
 - a kleine letters
 - A hoofdletters
 - i Romeinse cijfers in kleine letters
 - I Romeinse cijfers in hoofdletters



CSS heeft de voorkeur

Hoewel type in HTML5 een geldig attribuut is, moet het alleen worden gebruikt als de nummering van de lijst er echt toe doet, zoals bij een stappenplan, een technische handleiding of een juridische tekst. In andere gevallen heeft instellen met CSS de voorkeur. Daarvoor dient de eigenschap list-style-type. Zie hoofdstuk 11.

Oefening

Maak een nieuw HTML-bestand (of gebruik het HTML-bestand dat u eerder hebt gemaakt) en voeg daar de volgende code aan toe:

```
<p>Volg deze stappen:</p>
<ol>
  <li>Klik op Start.</li>
  <li>Klik op Doorgaan.</li>
  <li>Klik op Voltooien.</li>
  <li>Klik op OK.</li>
</ol>
```

Een nieuwe lijst begint weer bij 1. Het stappenplan kan na een alinea met uitleg doorgaan met stap 5 door het attribuut start de waarde 5 te geven:

```
<p>Als het misgaat doet u het volgende:</p>
<ol start="5">
  <li>Klik op Probeer opnieuw.</li>
  <li>Klik op Herstel fabrieksinstellingen.</li>
  <li>Klik op OK.</li>
</ol>
```

Oefening

Lijsten kunnen worden gecombineerd (ingesloten/genest) om meerdere niveaus te maken. Als voorbeeld maakt u een controlelijst in drie niveaus die bestaat uit een combinatie van (Romeinse) cijfers en letters. Hiervoor hebt u drie lijsten nodig:

- de eerste lijst bevat rubrieken met Romeinse cijfers;
- de tweede lijst bevat genummerde items;
- de derde lijst bevat subitems die met een letter worden aangeduid.

Probeer eerst deze lijst zelf te maken met dit stappenplan. Bekijk daarna pas de voorbeeldcode.

- 1 Open de eerste lijst en voeg een of meer items toe.
 - Sluit het laatste lijstitem niet af; de volgende lijst is een kind hiervan.
 - Sluit de lijst niet af.
- 2 Open de tweede lijst en voeg items toe.
 - Sluit het laatste lijstitem niet af; de volgende lijst is een kind hiervan.
 - Sluit ook deze lijst niet af.
- 3 Open de derde lijst en voeg items toe.
- 4 Sluit deze lijst nu af en sluit het omsluitende item af.
- 5 Voeg een volgend item toe; dit hoort bij de lijst op het tweede niveau.
- 6 Sluit ook deze lijst af en sluit ook weer het omsluitende item af.
- 7 Voeg een item toe; dit hoort bij het eerste niveau.
- 8 Sluit tot slot ook deze lijst af.

Typ als u de oefening niet zelf hebt voltooid de volgende code in uw HTML-bestand. Laat elk volgende niveau inspringen om het overzichtelijk te houden. Dergelijke geneste lijsten kunnen al snel onoverzichtelijk worden. Commentaar toevoegen kan ook helpen.

```
<ol type="I">
  <li>Stroomvoorziening <!-- item niet afsluiten, de volgende lijst hoort bij dit onderwerp -->
    <ol type="I">
      <li>Kabel <!-- item niet afsluiten, de volgende lijst hoort bij dit onderwerp -->
        <ol type="a">
          <li>Apparaataansluiting</li>
        </ol>
    </ol>
  </li>
</ol>
```

```
<li>Lichtnetaansluiting</li>
</ol> <!-- einde letterlijst -->
</li> <!-- einde item Kabel -->
<li>Aan-uitschakelaar</li>
</ol> <!-- einde numerlijst -->
</li> <!-- einde item Stroomvoorziening -->
<li>Toetsenbord <!-- vervolg Romeinse lijst; de volgende lijst hoort bij dit onderwerp -->
<ol>
  <li>Aansluiting</li>
</ol>
</li> <!-- einde item toetsenbord -->
</ol> <!-- einde Romeinse lijst -->
```

Merk op dat elke nieuwe lijst die u begint met `` automatisch telt vanaf 1 (of I of a, afhankelijk van het ingestelde type). In de vetgedrukte regel is het type toegevoegd voor de overzichtelijkheid; als u het weglaat wordt dezelfde nummering gebruikt. De rubriek Toetsenbord krijgt automatisch het juiste Romeinse cijfer II, omdat dit item een voortzetting is van de lijst die bovenaan is begonnen.

Ongeordende lijsten: ``

Een lijst waarvan de betekenis niet verandert als de volgorde wordt aangepast, is een **ongeordende lijst**. Deze wordt gemarkeerd met ``. Dit element heeft alleen de globale attributen.

Bij `` wordt het gebruik van het attribuut `type` afgeraden (het is *deprecated*). Bij een ongeordende lijst is het opsommingsteken decoratief en zegt het niets over de betekenis van de lijst. Gebruik hier altijd de CSS-eigenschap `list-style-type`.

Een ongeordende lijst heeft dezelfde opbouw als een genummerde lijst: open de lijst met ``, voeg items toe met `` en sluit de lijst af met ``. Ook lijsten op meerdere niveaus worden op dezelfde manier gemaakt.

Oefening

Verzin zelf een onderwerp en maak daarvan in het HTML-bestand een ongeordende lijst. Of neem de volgende code over en bekijk het resultaat in de browser.

```
<ul>
  <li>voorwiel
  <ul>
    <li>trommelrem</li>
    <li>naafdynamo</li>
  </ul>
```

```
</li>
<li>achterwiel</li>
<li>stuur</li>
<li>brappers</li>
</ul>
```

Oefening

De navigatie-elementen van een pagina zijn bij uitstek geschikt voor een ongeordende lijst. Het menu bestaat immers uit een opsomming van links en hoewel er logica in de volgorde zal zitten, blijft de betekenis hetzelfde als die volgorde verandert.

Bedenk zelf een navigatiemenu en codeer dat in het HTML-bestand. Welk structuurelement is speciaal voor navigatie bedoeld?

Probeer het eerst zelf en kijk daarna pas naar de volgende code:

```
<nav>
  <ul>
    <li><a href="index.htm">Home</a></li>
    <li><a href="producten.htm">Producten</a></li>
    <li><a href="diensten.htm">Diensten</a></li>
    <li><a href="service.htm">Klantenservice</a></li>
  </ul>
</nav>
```

Hiermee wint u geen designprijs, maar het gaat hier ook alleen om de structuur. In hoofdstuk 11 ziet u hoe een dergelijk menu wordt vormgegeven met CSS.

Kop boven een lijst

Het kan handig zijn om een lijst te voorzien van een kop of titel. Het lijstje van fietsonderdelen komt zonder kop nogal uit de lucht vallen. In het volgende codevoorbeeld is dat opgelost. Het zal u opvallen dat de kop niet is gemaakt met `<h1>` of een vergelijkbaar kopelement. De reden daarvoor is dat u met het element `<h2>` een nieuwe sectie begint. Dat kan de bedoeling zijn, maar vaak zal zo'n lijstje gewoon bij de omringende tekst horen.

```
<figure>
  <Figcaption>Kenmerken van onze fietsen</Figcaption>
  <ul>
    <li>voorwiel
    <ul>
      <li>trommelrem</li>
      <li>naafdynamo</li>
    </ul>
  </li>
</ul>
```

```
</ul>
</li>
<li>achterwiel</li>
<li>stuur</li>
<li>trappers</li>
</ul>
</figure>
```

Hier zijn de elementen `<figure>` en `<figcaption>` gebruikt om aan te geven dat de lijst een illustratie is bij de uitleg over aanbod van deze fietsverkoper en er een titel (bijschrift) boven te zetten. Hoe u deze elementen verder kunt gebruiken, leest u in de volgende paragraaf.

Illustraties en bijschriften: `<figure>` en `<figcaption>`

Niet om u te ontmoedigen, maar het correct gebruik van `<figure>` (illustratie) en `<figcaption>` (bijschrift) is nog best lastig. Vooraf:

- niet elke afbeelding is een `<figure>`;
- de elementen zijn niet alleen bedoeld voor afbeeldingen.

De volgende afbeeldingen zijn in elk geval geen `<figure>`:

- afbeeldingen die alleen dienen ter verfraaiing;
- logo's.



Logo in header

Een logo kan gewoon als `` worden opgenomen. Een gebruikelijk plaats is de header van de pagina. Het element ``, dat een verwijzing naar een afbeelding markeert, wordt besproken in hoofdstuk 5.

De reden om `<figure>` en `<figcaption>` aan HTML toe te voegen, is dat er nooit een duidelijke manier was om een bijschrift aan een figuur te koppelen. Met `<div>` en `` en CSS kon het visueel wel worden opgelost, maar niet semantisch. Deze nieuwe elementen hebben de echt de betekenis 'dit is een figuur' en 'dit is een bijschrift bij deze figuur'.

In de beschrijving van `<figure>` in de HTML 5-aanbeveling vindt u globaal het volgende: het element markeert flowcontent, eventueel met een bijschrift, die op zichzelf staat en waarnaar kan worden verwezen vanuit de hoofdinhoud. In dit verband betekent 'op zichzelf staand' niet dat het volledig losstaat van de hoofdinhoud. Het betekent wel dat het niet per se op een vaste positie in de tekst hoeft te staan waarin naar de illustratie wordt verwezen. Naast de tekst of in een bijlage kan ook.



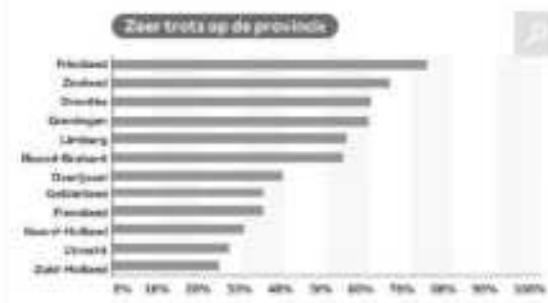
Friesland en Zeeland het meest trots op hun provincie. Onderstaan een deel van de voorpagina van de NOS over de resultaten van de enquête. Op pagina achtervolgt De Telegraaf dat de provincies Friesland en Zeeland ook het meest trots zijn op hun provincie. Tijdsblad AD was enthousiast over de NOS-enquête en de resultaten.

Afbeelding 3.2 Deze foto is versiering bij het artikel, maar voegt geen informatie toe. Daarom is het geen `<figure>` maar gewoon een ``.

Verwarring met het element `<aside>` is niet denkbeeldig, maar er is een belangrijk verschil: informatie in `<aside>` is slechts zijdelings gerelateerd aan de hoofdinhoud. Een `<figure>` hoeft weliswaar niet in de hoofdinhoud te staan, maar voegt daar wel betekenis aan toe.

Hoewel het voor de hand ligt om bij een figuur direct aan een afbeelding te denken, gaat het hier om illustraties in meer algemene zin. Dat kan een afbeelding zijn, maar ook een diagram, een foto, een tabel, een listing of – zoals in de paragraaf hiervoor bij de kenmerken van de fietsen – een opsomming. De codevoorbeelden in dit boek zijn typische voorbeelden van `<figure>`, evenals de afbeeldingen.

Met trots kunnen Friesland en Zeeland een belangrijke uitvoering: 14 procent van de Nederlanders noemt de Rode Kruiszone (L.F. en Z.L.) als hun provincie bestekend en trots in andere termen.



Om op te stellen is het verschil in kennis en ervaringen van mensen. Jongeren kennen vooral de historie en cultuur van de provincie. In het zuiden is dat de provinciale identiteit, in de noordelijke en westelijke provincies eerder de achtergrond in de achtergrond en achterhoede.

Afbeelding 3.3 Een schoolvoorbeeld van een `<figure>`: een grafiek die de cijfers uit het artikel overzichtelijk presenteert.

De conclusie dat elke afbeelding waar niet expliciet naar wordt verwezen ‘dus’ ook geen `<figure>` is, gaat echter weer te ver. Er hoeft niet letterlijk te staan: zie figuur 1. Een foto van een minister in overleg met de Tweede Kamer bij een verslag van dat overleg is wel degelijk een `<figure>`. Een foto waarop bijvoorbeeld is te zien dat de minister het moeilijk heeft, geeft extra betekenis aan het artikel. In code:

```
<article>
  <h1>Positie minister wankel</h1>
  <p>...artikeltekst zonder expliciete verwijzing naar de foto...</p>
  <figure> 
    <figcaption>De oppositie in de Tweede Kamer nam geen genoegen met de
    uitleg van de minister. Zijn positie staat ter discussie.</figcaption>
  </figure>
</article>
```



``

Het element ``, dat een verwijzing naar een afbeelding markeert, wordt besproken in hoofdstuk 5.

Afbeeldingen kunnen worden gegroepeerd om een overkoepelend bijschrift toe te voegen en daarbij elke afbeelding zijn eigen bijschrift te geven. Het volgende fragment is onderdeel van een artikel over het maken van chips:

```
<figure>
  <figcaption>Chips, van veld naar zak</figcaption>
  <figure> 
    <figcaption>De aardappels staan op het veld.</figcaption>
  </figure>
  <figure> 
    <figcaption>De aardappels worden verwerkt in de fabriek.</figcaption>
  </figure>
  <figure> 
    <figcaption>De zakken chips staan klaar voor de consument.</figcaption>
  </figure>
</figure>
```

In een fotoblog kunt u de artikelen bijvoorbeeld als volgt markeren:

```
<article>
  <header>
    <h1>Op het station in Milaan</h1>
    <p>Geplaatst op 31 mei 2019 door Reislustig</p>
```

Chips, van veld naar zak



De aardappels staan op het veld.



De aardappels worden verwerkt in de fabriek.



De zakken chips staan klaar voor de consument.

Afbeelding 3.4 Het eerste bijschrift staat als kop boven aan het beeldverhaal. De andere bijschriften staan onder de foto's.

```
</header>
<figure>
  <figcaption>Vorige week was ik Milaan. Het was druk op het station. Hoewel ik heen heb
  gevlogen, wilde ik tijdens de terugreis de romantiek van het spoor proeven.</figcaption>
</figure>
</article>
```

Op het station in Milaan

Geplaatst op 31 mei 2017 door Reishustig



Vorige week was ik Milaan. Het was druk op het station. Hoewel ik heen heb gevlogen, wilde ik tijdens de terugreis de romantiek van het spoor proeven.

Afbeelding 3.5 Een voorbeeld van een artikel in een fotoblog.

FAQ, metadata en definities: <dl>, <dt>, <dd>

Voor het markeren van veelgestelde vragen (FAQ), metadata of definities is een speciaal type lijst beschikbaar met de elementen <dl>, <dt> en <dd>. Het element <dl> (*description list*) markeert een lijst die bestaat uit paren gevormd door een naam en een waarde. De namen worden gemarkerd met <dt> (*description term*) en de waarden met <dd> (*description data*). Met andere woorden: het element <dd> (de waarde) zegt iets over het element <dt> (de naam).

Metadata kan bijvoorbeeld zijn gegevens over de auteur en redacteur van een artikel of, zoals voor in dit boek, gegevens over het boek. Dat kan er zo uitzien:

```
<dl>
  <dt>ISBN</dt>
  <dd>978-94-6356-081-8</dd>
  <dt>NUR</dt>
  <dd>994</dd>
  <dt>Trefwoorden</dt>
  <dd>HTML</dd>
  <dd>CSS</dd>
</dl>
```

Merk op dat één naam (Trefwoorden) meerdere waarden kan hebben. Het is niet toegestaan om binnen één lijst een naam meerdere keren te gebruiken voor verschillende definities.

Een lijst van veelgestelde vragen kan op vergelijkbare wijze worden gemarkerd:

```
<dl>
  <dt>Hoe markeer ik een kop?</dt>
  <dd>Met de elementen &lt;h1&gt; tot &lt;h6&lt;.</dd>
  <dt>Hoe markeer ik een alinea?</dt>
  <dd>Met het element &lt;p&gt;.</dd>
  <dt>Hoe markeer ik een FAQ?</dt>
  <dd>Met de elementen &lt;dl&gt;, &lt;dt&gt; en &lt;dd&gt;.</dd>
</dl>
```

Een lijst van afkortingen is ook een kandidaat en daarbij kunt u ook het element `<dfn>` gebruiken (zie de paragraaf *Speciale betekenis aangeven*).

```
<dl>
  <dt><dfn>HTML</dfn></dt>
  <dd>Hypertext Markup Language</dd>
  <dt><dfn>CSS</dfn></dt>
  <dd>Cascading Style Sheets</dd>
</dl>
```

Nieuw thema markeren: `<hr>`

Het element `<hr>` is in HTML 5 opnieuw is gedefinieerd. Oorspronkelijk was het bedoeld om een horizontale lijn (*horizontal rule*) te maken, maar omdat dat alleen presentatie is, is de definitie aangepast. `<hr>` markeert nu het begin van een nieuw thema op alineaniveau; het geeft het eind van de ene en het begin van de volgende sectie aan. In die zin lijkt het op `<section>`. De markeering met `<hr>` is echter wat subtieler. Het kan bijvoorbeeld in een verhaal worden gebruikt wanneer de vertelling zich verplaatst naar een andere locatie, een sprong maakt in de tijd enzovoort.

`<hr>` is een leeg element; u mag de zelfsluitende tag `<hr/>` gebruiken. Hoe de themawisseling wordt gevisualiseerd, bepaalt u met CSS. Het kan een lege regel zijn, een streepje, een sierlijk grafisch element enzovoort.

Lorem ipsum dolor sit amet, consectetur adipisicing elit.

Lorem ipsum dolor sit amet, consectetur adipisicing elit.



Lorem ipsum dolor sit amet, consectetur adipisicing elit.



Lorem ipsum dolor sit amet, consectetur adipisicing elit.

Afbeelding 3.6 Drie sectiemarkeringen met `<hr>`: een standaardlijn, een dikke lijn met achtergrondafbeelding en een lijn met een gekanteld lineair kleurverloop; allemaal CSS. en de technieken worden besproken in hoofdstuk 12.

Het laatste redmiddel: <div>

Het element <div> heeft geen speciale betekenis. Het wordt toegepast om inhoud (een reeks elementen) te groeperen zodat deze gemakkelijk is te benaderen voor opmaak of scripting. In de HTML 5-specificatie wordt <div> omschreven als uw laatste redmiddel, alleen te gebruiken wanneer alle hiervoor beschreven betekenisvollere elementen niet van toepassing zijn.

<div> heeft een rijke historie. Nadat paginaopmaak met tabellen in onbruik raakte, was <div> het element waarmee alle secties van een document werden gemarkerd. Nog steeds vindt u zonder moeite massa's websites met div's voor headers, footers, navigatiemenu's, kolommen en nog veel meer, herkenbaar aan constructies als <div id="header">, <div id="nav">, <div id="main">, <div id="article">, <div id="section">, <div id="footer">... Ziet u het patroon? Voilà HTML 5.

Dit gebruik is overigens niet per se fout. Een met <div> opgebouwde webpagina met het juiste DOCTYPE wordt gevalideerd als HTML 5 en zal de inhoud weergeven. Met CSS krijgt de inhoud vorm. Maar betekenis heeft de inhoud niet, althans niet voor de browser. Wilt u betekenisvolle HTML-documenten maken, toegankelijk voor bezoekers met beperkingen en die makkelijker te onderhouden zijn, dan gebruikt u de nieuwe HTML 5-elementen.

De rol van <div> is zeker niet uitgespeeld. Integendeel, veel websites worden gebouwd met UI-frameworks zoals jQuery of Bootstrap en die maken (buitensporig) veel gebruik van div's. Ook bij frameworks voor complete webapplicaties zoals Angular, React of Vue, worden gemakkelijk meer div's ingezet dan goed is voor de HTML, hoewel een ontwikkelaar die naast JavaScript ook HTML goed beheert daar een groot verschil kan maken. Voor zover het de lay-out betreft leert u in hoofdstuk 10, *Layouts maken met CSS*, hoe vooral CSS Grid Layout helpt om de hoeveelheid div's te beperken.

Speciale tekens in webpagina's

In de voorbeelden zijn al een paar keer de speciale tekens < en > voorbijgekomen. Maar er zijn natuurlijk meer speciale tekens, zoals letters met accenten (á en é) of tekens zoals & en €. U hebt eerder gezien dat het niet mogelijk is om in lopende tekst een element op te nemen:

<p>Een regelinde markeert u met het element <code>
</code>. Regelinde die u in de code-editor met Enter of Return invoegt, worden genegeerd.</p>

Zodra de browser punthaken tegenkomt, wordt de inhoud geïnterpreteerd als een tag. De voorgaande code wordt dan ook als twee regels weergegeven in de browser; er staat immers regeleinde in, de tag `
`.

Op de volgende manier wordt de tagnaam wel weergegeven in de browser:

`<p>Een regeleinde markeert u net het element <code>
</code>. Regeleinden die u in de code-editor met Enter of Return invoegt, worden genegeerd.</p>`

Hier is gebruik gemaakt van een benoemde tekenreferentie (*named character reference*). Er zijn ook decimale en hexadecimale tekenreferenties, maar in de meeste situaties kunt u prima uit de voeten met de benoemde referenties. Ze werken ook intuïtiever, omdat de naam van de referentie vaak is af te leiden uit de naam van het teken. Bijvoorbeeld de codes van de punthaken `<` en `>` staan voor *less than* (`lt`) en *greater than* (`gt`).

Een benoemde tekenreferentie is opgebouwd uit het teken & (de ampersand), de lettercode en een puntkomma. De lettercode is minimaal twee tekens lang en is hoofdlettergevoelig: zo geeft de code `à` een kleine letter à en `À` een hoofdletter Å. Nu zijn in het Nederlands hoofdletters met accented niet gebruikelijk, maar in andere talen wel. Daarbij maakt het ook uit voor niet-lettertekens. De code `|` geeft een enkele verticale lijn en `‖` een dubbele.

Van ASCII naar UTF-8

Welke tekens kunnen worden gecodeerd, is afhankelijk van de gebruikte tekenset. In de beginjaren van het web was dat ASCII. Elk teken is opgebouwd uit één byte (1 byte = 8 bits): 7 bits voor het teken en 1 controlebit. Daarmee kunnen slechts 128 tekens worden gemaakt (2 tot de macht 8), waarvan er ook nog 32 gereserveerd zijn voor andere doeleinden. Met de cijfers 0-9 en de letters a-z en A-Z plus speciale tekens zoals !, @, #, (en) is de koek op en lettertekens uit niet-Westerse talen passen er niet bij.

Daarna kwam de tekenset ISO-8859-1 en daarmee kunnen 256 tekens worden gecodeerd. Dat is nog steeds te weinig om meerderen talen te ondersteunen. Dat gemis is min of meer opgelost door subsets te ontwikkelen, maar die moesten in HTML 4 wel expliciet worden ingesteld.

Vanwege de beperkingen van de oude tekensets is de standaard Unicode ontwikkeld. Hiermee kunnen vrijwel alle (letter)tekens, interpunctie en symbolen ter wereld worden gecodeerd. In HTML 5 is de standaardtekenset UTF-8, wat staat voor Unicode Transformation Format. Een teken kan bestaan uit 1 tot 4 bytes. UTF-8 is compatibel met ASCII, want de eerste 128 tekens zijn hetzelfde.

Referenties gebruiken

Hoewel in veel gevallen de browser een &, % of ē goed zal weergeven, is het verstandig om elk teken dat geen cijfer van 0-9, een letter in het bereik a-z of A-Z of gangbaar leesteken is (spatie, komma, punt, puntkomma) door een tekenreferentie te vervangen. Op die manier weet u zeker dat het altijd goed gaat.

In veel gevallen kunt u (op basis van de Engelse naam) beredeneren wat de naam van de tekenreferentie zal zijn, maar om u op weg te helpen zijn enkele veelgebruikte codes opgenomen in de tabel. De ultieme naslag vindt u op internet: dev.w3.org/html5/html-author/charref. De pagina biedt een grafische weergave van de tekens met daarbij de benoemde, decimale en hexadecimale tekenreferenties.

Symbol	Code
>	>
<	<
&	&
©	©
€	€
é	´
è	è
ë	ë
{	&lcb;
}	&rcb;
vaste spatie	

Samenvatting

In dit hoofdstuk is het markeren van tekst en illustraties besproken.

- Voor koppen zijn er zes niveaus, van `<h1>` tot `<h6>`.
- Alineatekst wordt gemarkerd met `<p>`.
- Voor tekst met een speciale betekenis, bijvoorbeeld nadruk, een boektitel of een citaat, zijn elementen beschikbaar zoals ``, `<cite>` of `<q>`.
- Tekst kan met `<pre>` worden gegroepeerd met behoud van witruimte.
- Er kunnen geordende (met nummering/letters) of ongeordende (met opsommingstekens) lijsten worden gemaakt met `` en ``. Deze kunnen worden genest om verschillende niveaus aan te brengen. Elk item in de lijst is een element ``.
- Voor illustraties en bijschriften zijn er de elementen `<figure>` en `<figcaption>`. Niet elke afbeelding is een `<figure>` en een `<figure>` hoeft

geen plaatje te zijn, het kan ook een tabel, een grafiek, een kaart of citaat zijn, en meer.

- Voor FAQ's, metadata en definities zijn de lijselementen `<dl>`, `<dt>`, `<dd>`, en `<dfn>` beschikbaar.
- Speciale tekens zoals letters met accenten en `<>` en `&` kunnen het best worden gecodeerd met een benoemde tekenreferentie. Die is opgebouwd uit het teken `&` (de ampersand), de lettercode en een puntkomma, bijvoorbeeld `&nbsp`.

Het volgende hoofdstuk gaat over hyperlinks en andere koppelingen, bijvoorbeeld naar andere pagina's, naar bladwijzers op dezelfde pagina, van een miniatuur naar een grote afbeelding, naar een favicon enzovoort.

Oefeningen

- Maak een paginaheader met een kop en een tagline. Beoordeel of u in uw tekst kunt volstaan met losse koppen of dat `<header>`-elementen nodig zijn.
 - Bij het vorige hoofdstuk hebt u de hoofdstructuur van tijdschriftartikelen gemarkeerd. Gebruik nu artikelen om de markering op tekstdniveau aan te geven. Kooppen, alinea's en illustraties, maar ook benadrukte tekst, kleine lettertjes, super- en subschrift (`CO2`) enzovoort. Natuurlijk hoeft u de artikelen niet over te typen. U kunt de markeringen op papier aangeven of in het HTML-bestand dummytekst gebruiken.



Dummytekst maken

Er zijn websites die een hoeveelheid dummytekst op maat aanbieden, bijvoorbeeld lipsum.com. Die kunt u kopiëren en plakken. Nog handiger is dat het hulpmiddel Emmet (emmet.io, beschikbaar in Visual Studio Code) ook direct in uw HTML-bestand stukken lorem ipsum-teks kan maken. Typ `lorem` of `lipsum` en druk op Tab voor een standaardalinea tekst. Met een getal erachter maakt `lorem` zo veel woorden als u wilt. De opdracht `lorem100` maakt een tekst van honderd woorden.

- Bekijk of u elementen nodig hebt waarmee u speciale betekenis geeft aan woorden of delen van een zin, zoals `` of ``. Trap niet in de val om tekst 'alleen maar' cursief of vet te maken (zoals in een tekstverwerker), maar vraag u af welke speciale betekenis dat fragment heeft en kies daar de juiste tag bij. De opmaak komt later, met CSS.
- Gebruik tekenreferenties voor letters met accenten en andere speciale tekens.

- Woorden in een vreemde taal zijn in kranten en tijdschriften vaak cursief. Welke HTML-element zou u daarvoor gebruiken?
- Laat in de browser letterlijk de tekst € weergeven. Hoe pakt u dat aan?
- Maak verschillende soorten lijsten, op verschillende niveaus: genummerd, met letters, met bullets.
- Ga bij verschillende sites na of `<figure>` en `<figcaption>` zijn gebruikt. Beoordeel of het in die context een goede keus is.
 - Als `<figure>` niet is gebruikt maar wel de goede keus zou zijn, probeer dan de code te herschrijven.

Koppelingen maken

Dit hoofdstuk gaat over wat ooit het begin van HTML was: hyperlinks, ook koppelingen of kortweg links genoemd. Zonder links is internet een tijdschrift met rondom nietjes of rondom gebonden boek: verder dan de buitenant kom je niet. Links omvatten meer dan een mogelijkheid om te navigeren naar andere inhoud. Ze worden ook gebruikt om ondersteunende inhoud aan webpagina's toe te voegen, bijvoorbeeld stylesheets en JavaScript-bestanden.

U leert in dit hoofdstuk:

Wat hyperlinks zijn en hoe ze worden gemaakt met het element <a>.

Een hyperlink van een samenvatting naar een compleet artikel maken.

Een koppeling maken tussen een miniatuur en een grote foto.

Hoe u gebruiksvriendelijke hyperlinks schrijft.

Linken naar stylesheets, favicons en andere externe bestanden.

Inleiding

De hypertext uit de Hypertext Markup Language bestaat dankzij hyperlinks, zo eenvoudig is het. En zonder hypertext was er geen world wide web. Een *hyperlink* verbindt informatie, u navigeert ermee naar een andere plaats in hetzelfde document, naar een ander document op dezelfde of een andere webserver, naar een grote versie van een foto of naar een download. Naast hyperlinks zijn er ook *links naar externe bronnen* die het document ondersteunen, bijvoorbeeld stylesheets of scripts.

In dit hoofdstuk komen beide soorten links aan bod, waarbij wordt ingegaan op de elementen `<a>` en `<link>`.



Het element `<area>`

Bekijk voor een beschrijving van het element `<area>` (dat ook in de categorie links valt) in hoofdstuk 5 de paragraaf *Klikbare gebieden: usemap*.

Overheid.nl

enigalijzer over informatie en diensten van alle overheden

MijnOverheid.nl

The screenshot shows the Overheid.nl website. At the top, there's a navigation bar with 'Overheid.nl' and 'MijnOverheid.nl'. Below the navigation is a search bar. The main content area features four large cards:

- Aankondigingen over uw buurt:** Shows a photo of a community meeting and text about local planning and traffic regulations. Links include 'Berichten over uw buurt' and 'Aanmeldingsreeks'.
- Dienstverlening:** Shows a photo of people at a service counter and text about permits, declarations, and applications. Links include 'Inzake', 'Levensaanpassingen', and 'Ondernemerschap'.
- Beleid & regelgeving:** Shows a photo of a document and text about policy publications. Links include 'Wettenspraak', 'Wet', 'Bewijsmateriaal', 'Rechtspraak', 'Intercommissie', and 'Alle beleid & regelgeving'.
- Contactgegevens overheden:** Shows a photo of a city skyline and text about contact details for government organizations. Links include 'Gebouwen', 'Bureaus', 'Wetenschappen', 'Milieuzaken', and 'Alle overheidsovereenkomsten'.

Afbeelding 4.1 De startpagina van overheid.nl bestaat grotendeels uit hyperlinks.

Verbindingen leggen met

Het is even briljant als eenvoudig: u klikt 'ergens' op en er verschijnt 'iets anders' in beeld: een andere pagina, een ander stuk tekst, een grote foto, een filmpje, een pdf-pagina enzovoort. Wat er ook verschijnt, uw vervoermiddel is altijd een hyperlink, kortweg link of koppeling. In de oertijd was een hyperlink (bijna) altijd herkenbaar aan blauwe, onderstreepte tekst, maar links kunnen op het moderne web allerlei vormen hebben. Welke vorm dat is, wordt in de HTML-specificatie niet gezegd; die gaat immers alleen over de betekenis van inhoud. Daarbuiten zijn er wel richtlijnen voor herkenbaarheid van hyperlinks en bezoekers hebben ook een bepaalde verwachting van het uiterlijk van koppelingen, maar niemand verplicht u tot een bepaalde vorm.



Afbeelding 4.2 Sommige bedrijven danken hun bestaan aan hyperlinks.

Een hyperlink wordt gemaakt met het element . Een sluittag is verplicht en de inhoud van het element is de klikbare inhoud van de link:

```
<a attributen>...klikbare inhoud...</a>
```

De *klikbare inhoud* in deze coderegel mag u ruim zien. Inhoud is niet alleen een tekstfragment. Het kan een afbeelding zijn, een kop, een alinea of zelfs een heel artikel. Bijna alles kan in een hyperlink worden opgenomen, behalve een andere hyperlink of een button (interactieve elementen). U kunt dus niet een hyperlink maken van een heel artikel als daarin een andere koppeling staat, want dan zou u een hyperlink in een hyperlink opnemen. Op een blog met korte voorpagina-artikelen die met *Lees meer...* verwijzen naar het

complete artikel, kunt u heel goed in plaats van alleen de Lees meer-koppeling van het hele artikel een hyperlink maken naar het volledige verhaal.

Attributen van <a>

Attributen zorgen ervoor dat het element ook daadwerkelijk als hyperlink kan functioneren. Behalve de globale attributen (zie hoofdstuk 2) kunt u bij <a> ook de volgende attributen gebruiken:

- **href** Dit attribuut bevat de URL van de hyperlink. Zonder href kan de gebruiker klikken tot hij een ons weegt, maar zal er niets gebeuren. Soms is dat precies wat u wilt. In een navigatiemenu is het bijvoorbeeld nuttig als de link naar de huidige pagina niet werkt, zoals in het volgende voorbeeld waarin de bezoeker zich bevindt op de pagina over cavia's:

```
<nav>
  <a href="home.htm">Home</a>
  <a>Cavia's</a>
  <a href="parkiet.htm">Parkieten</a>
  <a href="katten.htm">Katten</a>
</nav>
```

U ziet hier zowel hoe href wordt gebruikt als waarom het nuttig kan zijn om het weg te laten.



Href weglaten is beter dan leeg laten?

Zonder href is een element <a> geen hyperlink en gebeurt er bij klikken niets. Maar als href leeg is, is <a> gewoon een hyperlink en wordt de pagina opnieuw geladen. Om dat te voorkomen kan href bij de link naar de huidige pagina beter worden weggelaten.

De links in dit voorbeeld leiden naar pagina's op dezelfde website en zijn relatieve URL's (zie hoofdstuk 5). Om een link naar een andere website te maken hebt u een volledige (absolute) URL nodig:

```
<a href="https://www.andere-website.nl">Naar de andere website</a>
```

Een link naar een specifieke pagina op die andere website ziet er zo uit:

```
<a href="https://www.andere-website.nl/pagina.html">Naar de pagina op de andere website</a>
```

Bedenk dat automatisch de startpagina (*homepage*) index.html wordt geopend als u alleen de domeinnaam opgeeft.



Startpagina is altijd index.html

Elke webserver zoekt als u een website aanroeft naar het bestand index.html, index.htm of index.php. Als u in de browser het adres www.een-website.nl typt, stuurt de webserver altijd de pagina terug die wordt gevonden met <https://www.een-website.nl/index.html>. Een slecht ingestelde webserver zal wanneer het bestand index.html ontbreekt een bestandenlijst (zoals in Verkenner) naar de browser sturen. Dit is een enorm beveiligingslek. Repareer dat zo snel mogelijk of spreek uw webhoster erop aan.

- **target** Hiermee wordt bepaald waar het doel van de link wordt geopend. Wanneer u dit attribuut weglaat of instelt op de waarde _self, wordt de huidige pagina in de tab of het venster van de browser vervangen door het doel van de link.
 - Met de waarde _blank wordt het doel van de link geopend in een nieuw venster of nieuw tabblad.
 - Er zijn ook nog de waarden _parent en _top, waarbij het doel wordt geopend in het bovenliggende document (*parent*) of het document boven in de hiërarchie (*top*). Bij links in een *<iframe>* met het attribuut sandbox kan het resultaat anders zijn, omdat dan diverse navigatiemogelijkheden (kunnen) zijn uitgeschakeld.



Webpagina in een webpagina

Met *<iframe>* kan externe inhoud in een pagina worden getoond. Zie hoofdstuk 5.

- **download** Het attribuut download geeft aan dat de koppeling niet is bedoeld voor navigatie, maar om het doel in het attribuut href te downloaden. download kan een waarde hebben die een naam bevat waarmee de download kan worden opgeslagen. Het bestandssysteem voegt zelf de juiste extensie toe. Deze eigenschap werkt niet op het lokale bestandssysteem; als u het wilt testen, moeten de bestanden op een webserver staan. In het voorbeeld wordt nadat op de link is geklikt het venster Opslaan als geopend, met daarin als voorgestelde bestandsnaam zonnebloem.png (de bestandsextensie wordt overgenomen uit het doelbestand).

```
<a href="/images/WRK2019-08-02-127.png" download="Zonnebloem.png">Download eer zonnebloem!</a>
```

- **hreflang** Hiermee wordt aangegeven in welke taal het doel is geschreven (als het een andere is dan de taal van de huidige pagina). Dit kan van pas komen als er verschillende taalversies van een pagina zijn. Dan wordt hreflang gebruikt in combinatie met het attribuut rel="alternate", zie

hierna. Een link naar een Engelse versie van dezelfde pagina zou er zo uit kunnen zien:

```
<a href="index_en.htm" >reflang="en" rel="alternate">English versions</a>
```

- **type** Dit attribuut kan worden toegevoegd om het MIME-type van het doelbestand aan te geven. In het verleden was het belangrijk om zo scripts en stylesheets goed te laden, maar tegenwoordig hebt u het nog zelden nodig. De meeste bestandstypen worden automatisch goed verwerkt.
- **rel** Dit attribuut geeft welke relatie het doel en het document hebben. Dit attribuut heeft nogal wat mogelijke waarden, die ook gelden voor het element `<area>`, maar die u niet vaak nodig zult hebben. Een compleet overzicht staat op www.w3.org/TR/html5/links.html#linkTypes. Met `rel=stylesheet` wordt een stylesheet aangeduid. De waarde `author` geeft aan dat de link informatie over de auteur bevat. (Kan worden gecombineerd met `mailto:to`; zie de paragraaf *Link naar een e-mailadres*.)
- **rev** Het attribuut `rev` staat voor *reverse link*; het geeft aan waar de link vandaan komt. Dat kan de homepage zijn, maar ook een andere locatie in een document.



Geen onmisbare info in de titel

Het globale attribuut `title` kan worden gebruikt voor een tooltip met een beschrijving van de link. Aangezien die tooltip alleen zichtbaar wordt als u een muis gebruikt (en dus niet wordt getoond op mobiele apparaten of bij toetsenbordnavigatie) wordt geadviseerd geen voor de gebruiker onmisbare informatie in een attribuut `title` te zetten.

Table of Contents

1 Introduction
1.1 Background
1.2 Audience
1.3 Scope
1.4 History
1.5 Design notes
1.5.1 Serializable of script execution
1.5.2 Compliance with other specifications
1.5.3 Extensibility
1.6 HTML vs XHTML
1.7 Structure of this specification
1.7.1 How to read this specification
1.7.2 Typographic conventions
1.8 Privacy concerns
1.9 A quick introduction to HTML
1.9.1 Writing secure applications with HTML
1.9.2 Common pitfalls to avoid when using the scripting APIs
1.9.3 How to catch mistakes when writing HTML: validators and conformance checkers
1.10 Conformance requirements for authors
1.10.1 Presentation-level markup
1.10.2 Syntax errors
1.10.3 Restrictions on content models and on attribute values
1.11 Suggested reading

Afbeelding 4.3 De inhoudsopgave van een omvangrijk webdocument bestaat natuurlijk uit hyperlinks.

Bladwijzers maken

Van de globale attributen vervult `id` in relatie tot `<a>` een speciale rol. Het maakt het mogelijk om een bladwijzer te maken, een ankerpunt in de pagina waar met een hyperlink naartoe kan worden gesprongen. De inhoud met de bladwijzer wordt boven aan het browservenster geplaatst. U kunt naar een bladwijzer op dezelfde pagina, maar ook naar een andere pagina en uiteraard naar een andere website. Er zijn twee dingen voor nodig: een element met een `id` en een link die daarnaar verwijst.



Drie dingen

Eigenlijk is er nog een derde voorwaarde: de pagina moet lang genoeg zijn om het element met de bladwijzer boven aan de pagina te kunnen plaatsen. Is dat niet zo, dan gebeurt er ogenschijnlijk niets. Beetje flauw, maar toch een mogelijke instinker. Maak zo nodig het browservenster smaller om een bladwijzer te kunnen testen.

De bladwijzer wordt als volgt gemaakt:

```
<h1 id="bladwijzer">Bladwijzers maken</h1>
<p>Zo maakt u een bladwijzer.</p>
```

Een hyperlink verzorgt de navigatie naar deze bladwijzer. In de link wordt die aangeduid met het hekje (#):

```
<p>Met een <a href="#bladwijzer">bladwijzer</a> wordt naar een specifiek punt in de pagina genavigeerd.</p>
```

Een bladwijzer op een andere pagina op dezelfde wordt u als volgt gemaakt:

```
<p>Op de pagina Markeringen vindt u meer informatie over
<a href="markeringen.html#bladwijzer">bladwijzers.</a>
```

Bladwijzers op door anderen beheerde websites kunt u uiteraard niet beïnvloeden, maar ernaar linken is natuurlijk geen probleem. Kopieer en plak de externe URL in het element `<a>`:

```
<p>Als het element <a href="https://www.w3.org/TR/html52/textlevel-semantics.html#the-a-element">
<code>&lt;a href="https://www.w3.org/TR/html52/textlevel-semantics.html#the-a-element"></code></a> een attribuut <a href="https://www.w3.org/TR/html52/links.html#attrdef-a-href"><code>href</code></a> heeft, is het een hyperlink.</p>
```

De koppelingstags zijn hier vet en de linktekst is cursief.



Let op typefouten

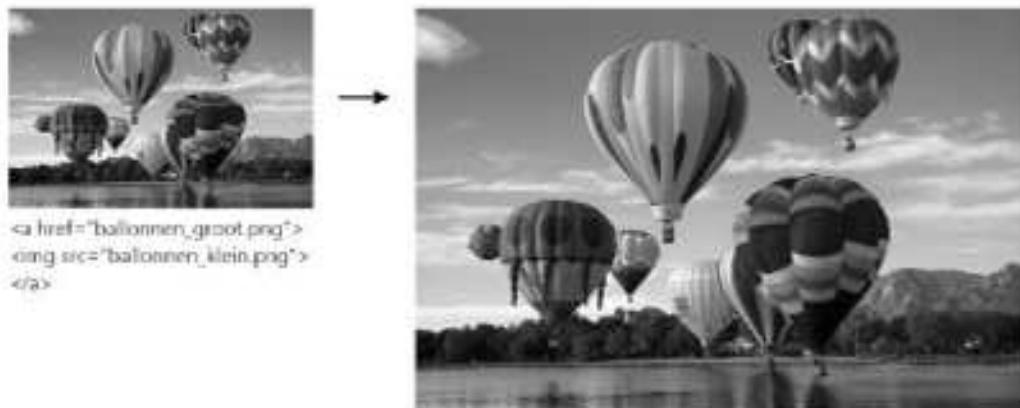
Werkt een bladwijzer niet? Een typefout is snel gemaakt en hoofdletters en kleine letters moeten overeenkomen. ARTIKEL2 is wat anders dan artikel2.

Link naar een grote foto

U kent dat wel: een site toont foto's in klein formaat en als u erop klikt wordt het scherm gevuld met de grote versie van de afbeelding. Vaak wordt daarbij JavaScript gebruikt om speciale effecten te creëren, bijvoorbeeld een fotovenser dat de onderliggende pagina afdekt en schaduweffecten, ook wel *overlay* genoemd. Nu kunt u met alleen CSS ook al mooie effecten bereiken (zie hoofdstuk 8 en verder) en zelfs een overlay zonder JavaScript is mogelijk, maar het gaat hier om de kern en dat is gewoon een hyperlink.

We nemen als voorbeeld één foto uit een pagina met miniaturen. (Zie hoofdstuk 5 voor uitleg over het afbeeldingelement ``.) Het enige wat u hoeft te doen is een linkelement om de foto plaatsen:

```
<a href="ballonnen_groot.png">  
    
</a>
```



Afbeelding 4.4 De kleine foto is een hyperlink naar de grote.

Zo eenvoudig is het: het element `` toont de kleine afbeelding en de hyperlink zorgt ervoor dat de grote afbeelding wordt opgehaald en getoond in dezelfde pagina. Met de knop Terug kan de bezoeker de oorspronkelijke pagina opnieuw openen. U kunt natuurlijk ook een link naar de oorspronkelijke pagina toevoegen.

Link naar het vervolg van een artikel

Op een nieuwssite of elke andere site met diverse berichten op de pagina worden op de homepage zelden de volledige versies van de artikelen getoond.

Vaak blijft het bij de kop en een intro of een samenvatting. Met een hyperlink met de tekst 'Lees het volledige artikel' of door van de kop of alle tekst een hyperlink te maken, kan de hele complete verhaal worden getoond.

De eerste optie is eenvoudig zo iets als dit:

```
<p><a href="artikelen/artikel1.htm">Lees het volledige artikel.</a></p>
```



Afbeelding 4.5 ‘Lees meer’ is eigenlijk achterhaald. Met HTML5 kan de hele samenvatting linken naar het volledige artikel.

Bij de kop van een artikel gaat het op dezelfde manier:

```
<article>
<h2><a href="/">Team Rond wint WK ballonvissen</a></h2>
<p>Het wereldkampioenschap ballonvissen is verrassend gewonnen door het Nederlandse team Rond.</p>
</article>
```

Op de volgende manier wordt kan van het hele artikel (inclusief foto) een hyperlink gemaakt:

```
<article>
<a href="artikel.htm">
<h2>Team Rond wint WK ballonvissen</h2>

<p>Het wereldkampioenschap ballonvissen is verrassend gewonnen door het Nederlandse team Rond.</p>
</a>
</article>
```

```
</a>  
</article>
```

Merk op dat de kleine foto nu geen link heeft naar de grote versie; binnen een hyperlink mogen geen andere hyperlinks worden opgenomen. De foto linkt wel – net als de rest van het artikel – naar de volledige versie van het verhaal. Op die pagina kan de link naar de grote foto wel worden gebruikt en dat ligt ook meer voor de hand. Wie de moeite neemt om door te klikken naar het hele artikel, zal meer interesse hebben voor de grote foto.

Link naar een e-mailadres

Rechtstreeks vanuit de browser een e-mail versturen met alleen HTML kan niet, zelfs niet met HTML 5. U kunt wel een link met een e-mailadres aanbieden. Klikt de bezoeker daarop, dan wordt zijn mailprogramma (Outlook, Mail) geopend, mits hij dat heeft geïnstalleerd. In het vak Aan is dan het adres uit de e-maillink al ingevuld.

De link bestaat uit `mailto:` (merk op dat hierbij geen `//` wordt gebruikt) en het e-mailadres:

```
<a href="mailto:reacties@handboek-html-css.nl">Mail de auteur.</a>
```

Met het attribuut `rel` wordt de link nog duidelijker:

```
<a href="mailto:reacties@handboek-html-css.nl" rel="author">Mail de auteur.</a>
```

U kunt zelfs het onderwerp al bepalen door het adres uit te breiden met `?subject=`:

```
<a href="mailto:reacties@handboek-html-css.nl?Subject=Vraag over hyperlinks">Mail de auteur.</a>
```

De beperking is dat de bezoeker een mailprogramma heeft geïnstalleerd en niet zijn browser (webmail) gebruikt.

Link naar een telefoonnummer

Met een groot aantal mobiele bezoekers is een telefoonlink erg handig. Ook die wordt gemaakt met behulp van het attribuut `href`, waarin het protocol `tel:` en het telefoonnummer worden opgenomen:

```
<a href="tel :+310621436587">Bel 06-21436587</a>
```

De link ziet eruit als een gewone hyperlink (afhankelijk van wat u ervan maakt met CSS), maar opent op een mobiele telefoon de telefoonapp met het nummer al ingevuld. De potentiële klant hoeft alleen nog op de belknop te tikken.

Een pdf-bestand weergeven of downloaden

Het aanbieden van pdf-bestanden is een veelgebruikte toepassing. Daarbij zijn er twee opties:

- lezen in de browser;
- downloaden naar het apparaat.

Met het element `<a>` kunnen beide opties worden gebruikt. Lezen in de browser gaat bijna vanzelf, maak een hyperlink naar het pdf-bestand:

```
<p>Lees de <a href=".pdf/h05.pdf" >pdf van hoofdstuk 5</a>.</p>
```

Het bestand wordt meestal probleemloos en direct geopend in de browser. Mocht er geen invoegtoepassing voor de pdf zijn geladen, dan wordt de bezoeker daar op gewezen.

Om het bestand te kunnen downloaden moet de link iets worden aangepast:

```
<p>Lees de <a href=".pdf/h05.pdf" download="hoofdstuk 5">pdf van hoofdstuk 5</a>
```

Het attribuut `download` heeft hier twee functies: het voorkomt dat de pdf direct wordt geopend in de browser en het biedt aan het bestand op te slaan met een handige naam.

Klik hier voor richtlijnen

Voor goede, bruikbare hyperlinks is meer nodig dan de juiste HTML-markering. Belangrijk is ook dat de hyperlinks herkenbaar zijn (opmaak) en dat de tekst het doel van de link voldoende duidelijk maakt.

De ingebouwde stylesheet van browsers geeft hyperlinks sinds jaar en dag blauw en onderstreept weer. Een bezochte link is paars en een link waarop wordt geklikt, licht rood op. Moet u dat zo laten? Niet perse. Maar het is wel raadzaam dat hyperlinks opvallen en dat tekst niet wordt onderstreept als het geen hyperlink is, zelfs niet als op de pagina geen hyperlinks met onderstrepen worden gebruikt. U brengt bezoekers ermee op een dwaalspoor. Aanwijseffecten zijn leuk om te maken, maar alleen bezoekers met een muis hebben er wat aan en dat zijn er steeds minder. Dan nog zullen de hyperlinks in

rusttoestand moeten opvallen, terwijl het uw bedoeling is dat de bezoeker met zijn muis de hele pagina scant...

Hetzelfde geldt voor grafische pagina's, waar tekst een ondergeschikte rol speelt. Wel zal de oplossing anders zijn dan op een tekstgeoriënteerde pagina. CSS (eventueel aangevuld met JavaScript) biedt ruim voldoende mogelijkheden om van de navigatie geen zoek-de-verschillenpuzzel te maken.

Naast het uiterlijk is ook de inhoud van hyperlinks belangrijk. Laat u in elk geval niet verleiden tot het nietszeggende 'klik hier', want die twee woorden verschaffen geen enkele informatie over het doel van de link en 'hier klikken' lukt sowieso niet op mobiele apparaten en screenreaders. Er is op internet veel informatie te vinden over het schrijven van goede hyperlinkteksten. Samengevat komt het neer op het volgende:

- Gebruik om de bezoeker tot actie aan te zetten korte, maar betekenisvolle linktekst die:
 - informatief is wanneer deze buiten de context wordt gelezen;
 - uitlegt wat de link te bieden heeft;
 - geen technisch verhaal bevat over hoe de link werkt;
 - geen werkwoord bevat.

Hiermee valt deze dus af:

[Klik hier](#) om onze recepten voor cupcakes (ook glutenvrij) te bekijken.

Hoe dan wel? De oplossing is vaak om de links niet geïsoleerd te bekijken, maar als onderdeel van de hele tekst. Om bij het voorbeeld te blijven: op deze website met passie voor cupcakes gaat aan de link zeker een verhaal vooraf over het maken van cupcakes en hoe lekker ze ook kunnen zijn zonder gluten. In plaats van de zin hiervoor als afsluiter van het verhaal te gebruiken, kunnen de links beter in de tekst worden opgenomen:

Met onze [recepten](#) maak je zelf de heerlijkste cupcakes. We hebben ook [recepten zonder gluten](#).

Zo zou het ook kunnen:

Bekijk onze [recepten voor cupcakes](#). We hebben ook [glutenvrije cupcakerецепты](#).

Nog wat voorbeelden. Niet zo:

Bekijk mijn vakantiefoto's [hier](#), [hier](#) en [hier](#).

maar zo:

Foto's van mijn vakantie: in het hotel, op het strand en in het oerwoud.

DigiToegankelijk.nl is de centrale plek voor informatie over digitale toegankelijkheid. Je vindt hier praktische uitleg over de toegankelijkheidseisen en alles over wet- en regelgeving.

Afbeelding 4.6 Een fragment van de overheidswebsite www.digitoe-gankelijk.nl. Zo schrijft u hyperlinks.

Verder waarderen bezoekers het als ze niet worden verrast. Maak duidelijk dat een link direct naar een download leidt. In het voorbeeld van de vakantiefoto's zou de link kunnen leiden naar een pagina met miniaturen die doorlinken naar de grote foto's. De bezoeker kan dan zelf kan besluiten of hij het de moeite waard vindt om de grote foto's te downloaden en bekijken.



Webrichtlijnen

Meer uitleg over het maken van goede hyperlinks (en het bouwen van toegankelijke websites) vindt u bijvoorbeeld op de site www.digitoe-gankelijk.nl.

Schrijf een linktekst waaruit het doel van de link duidelijk af te leiden is

Ook zonder context moet het duidelijk zijn waar links naar verwijzen. Een linktekst als 'klik hier' of 'lees verder' is niet duidelijk als je de omliggende tekst of de visuele presentatie niet ziet. Schrijf daarom linkteksten die op zichzelf al duidelijk aangeven waar ze naartoe zullen leiden.

Hoe pas ik dit toe?

Denk je voor dat je een linktekst heel zonder dat je de rest van de webpagina kunt zien of lezen. Waar je dan nog steeds waar de link heen zal leiden? Als dat zo is, dan is het een goede linktekst.

Afbeelding 4.7 Digitaaltoegankelijkheid.nl biedt veel nuttige informatie, niet alleen over goede hyperlinks.

Externe bestanden koppelen met <link>

Het element <link> is net als <a> en <area> bedoeld om contact te maken met externe bronnen, maar in dit geval gaat het om metadata: informatie over de informatie. Met link koppelt u extra informatie aan het document. Het bekendste voorbeeld is een stylesheet. Met <link> kan echter meer dan dat, bijvoorbeeld een pictogram instellen voor in het browservenster (*favicon*).



Afbeelding 4.8 Favicons op websites van Nederlandse kranten.

De attributen href en rel zijn verplicht. href bevat de URL naar de externe bron en rel geeft aan welke relatie er met de bron is. <link> kan alleen worden gebruikt in het element <head> van het document.

Stylesheet koppelen

De opmaak van een HTML-document wordt vastgelegd in een of meer stylesheets. Het is wel steeds gebruikelijker geworden om alle opmaak in één stylesheet te schrijven, maar er kunnen ook verschillende stylesheets zijn, bijvoorbeeld voor speciale opmaak zoals hoog contrast of voor een printversie van de pagina met een afzonderlijke lay-out. Met het attribuut media wordt aangegeven voor welk type apparaat en voor welke schermgrootte een stylesheet bestemd is. Meer over *media queries* vindt u in hoofdstuk 10, want het onderwerp staat dichter bij CSS dan bij HTML.



Liever één stylesheet

U kunt verschillende stylesheets aanbieden, maar uit oogpunt van efficiënt netwerkgebruik kunt u meestal beter één grote dan veel kleintjes hebben. Deze ene stylesheet bevat de media queries waarmee wordt bepaald welke opmaak van toepassing is. Meer hierover in hoofdstuk 10. De reden is dat alle stylesheets worden geladen, ook de alternatieven, zodat ze klaarstaan voor gebruik. De browser weet vooraf immers niet wat hij nodig zal hebben. Elke aanvraag van de browser naar de webserver (*HTTP-request*) kost tijd en bandbreedte. Zeker voor mobiele apparaten geldt hoe minder aanvragen, hoe beter.

Terug naar <link>. De koppeling met de standaardstylesheet heeft niet meer nodig dan de verplichte attributen rel en href:

```
<link rel="stylesheet" href="stijlen.css">
```

Is de opmaak verdeeld over meerdere stylesheets, dan laadt u die op dezelfde manier:

```
<link rel="stylesheet" href="meerstijlen.css">
```

Alternatieve stylesheets moeten naast href en rel ook het attribuut title hebben. Daarmee worden ze weergegeven in het menu van de browser. Tenminste, als de browser zo'n menu heeft. In Firefox vindt u de stylesheetkeuze in het menu Beeld, Paginastijl en dat is het dan wel. Chrome en Edge bieden geen toegang tot alternatieve stylesheets. Het is dan ook veel beter een in JavaScript geschreven stylesheetwisselaar aan te bieden (op internet vindt u diverse mogelijkheden). Met een goed herkenbare knop biedt u bezoekers dan de mogelijkheid een versie van uw pagina met hoog contrast of extra grote tekst te laden.

Een favicon gebruiken

Een favicon is een pictogram waaraan de webpagina is te herkennen in het tabblad van de browser (of op een tablet, smartphone of als tegel in Windows). Nu kan zo'n pictogram nogal klein zijn en dan is het maken ervan priezelwerk. Er zijn speciale (online) editors voor, maar ook bijvoorbeeld Photoshop is een prima stuk gereedschap. Houd er rekening mee dat u letterlijk op pixelniveau moet werken, want klein wil zeggen vanaf 16 bij 16 pixels voor een browsertab.



Afbeelding 4.9 Voor het maken van favicons zijn online editors beschikbaar, zoals x-icon. Een goed alternatief is realfavicongenerator.net.

Voor het toevoegen van zo'n pictogram gebruikt u de waarde `rel="icon"`. Dit kan worden aangevuld met het attribuut `sizes`, dat aangeeft in welke afmetingen het pictogram beschikbaar is. Het weergaveapparaat kan dan zelf de beste optie kiezen. De standaardtoepassing voor een pictogram in de browser-tab is:

```
<link rel="icon" sizes="16x16 24x24 32x32 64x64" type="image/png" href="favicon.png">
```

Deze code zorgt ervoor dat in de tabbladen van Chrome en Firefox een favicon wordt getoond, maar niet in Internet Explorer. Voordat u zich verdiept in alle mogelijke code om voor diverse apparaten faviconlinks te maken: u kunt ook volstaan met een bestand `favicon.ico` in de root (de hoofdmap) van de website.

Op dit moment ondersteunen alle desktopbrowsers afbeeldingen in het bestandsformaat PNG. Dit type bestand kan met elke bitmapeditor worden gemaakt. Voor de beste compatibiliteit kan echter beter het formaat ICO worden gebruikt. Een `.ico`-bestand is een containerbestand dat afbeeldingen in verschillende formaten kan bevatten. Het biedt bovendien het voordeel dat u het in de root van de website kunt plaatsen (zie hierna). Noem het bestand `favicon.ico`, ook weer vanwege maximale uitwisselbaarheid tussen browsers. PNG-bestanden kunnen met (online) iconeditors worden omgezet naar ICO.

Het onderwerp heeft al heel wat mensen langdurig van de straat gehouden en de informatie die zij hebben verzameld kan u verder op weg helpen. Een bron is github.com/audreyr/favicon-cheat-sheet. Deze pagina bevat ook links naar editors en andere bronnen over dit onderwerp. Aanwijzingen voor het maken van favicons zijn bijvoorbeeld te vinden in het artikel op www.creativebloq.com/illustrator/create-perfect-favicon-12112760.



Favicon is een hardnekkig bestand

Een eenmaal geladen favicon waar u vanaf wilt, kan in Google Chrome/Canary bijzonder hardnekkig zijn. De enige bekende oplossing is een paardenmiddel, maar het werkt: typ in Windows in het venster Uitvoeren de opdracht %appdata%. Navigeer naar Local\Google\Chrome\User Data\Default en hernoem of verwijder het bestand favicon. In OS X moet u zijn in de user library en de map Application Support/Google/Chrome/Default/.

Opeenvolgende pagina's

Soms vormen webpagina's een logische reeks, bijvoorbeeld in een stappenplan of een fotogalerij. Ook opeenvolgende productpagina's of andere overzichtspagina's kunnen een vaste volgorde hebben. Deze logica kunt u vast-

leggen in het element `<link>` met het attribuut `rel="next"` en `rel="prev"`. Hiermee worden de volgende (`next`) en de voorgaande (`prev`) pagina gemarkeerd. In het volgende voorbeeld is `pagina2.htm` geopend, waardoor pagina 1 de voorgaande en pagina 3 de volgende pagina is:

```
<link rel="prev" href="pagina1.htm">
<link rel="next" href="pagina3.htm">
```

Op pagina 1 gebruikt u alleen de waarde `next` (er is geen voorgaande pagina):

```
<link rel="next" href="pagina2.htm">
```

Op de laatste pagina komt alleen `prev` (er is geen volgende):

```
<link rel="prev" href="pagina9.htm">
```

Zoekmachines zoals Google en Bing kunnen de informatie gebruiken om bezoekers naar de juiste pagina te sturen. Daarmee zijn `next` en `prev` vooral nuttig in het kader van zoekmachineoptimalisatie, een onderwerp waarover online veel informatie is te vinden.

Vooraf laden: prefetch, preload, prerender en dns-prefetch

Het kan nuttig zijn om inhoud alvast klaar te zetten voor de bezoeker, tenminste als het waarschijnlijk is dat die inhoud zal worden bekeken. De bezoeker hoeft dan na het activeren van de link niet of minder lang te wachten totdat een grote afbeelding of de volgende pagina in de browser wordt getoond. Dit vooraf laden wordt ingesteld met de waarde `prefetch` van het attribuut `rel`. De waarde van `href` geeft aan welke bron moet worden geladen. Leden wil zeggen dat de bron alvast wordt opgeslagen op het apparaat van de bezoeker, in de browsercache.

```
<link rel="prefetch" href="grote-afbeelding.png">
<link rel="prefetch" href="volgende.html">
```

Nu wordt bij `prefetch` én alleen de genoemde bron opgehaald én blijft het een gok of de bezoeker die bron zal willen laden. Bij een HTML-pagina wordt alleen de HTML opgehaald en moeten de afbeeldingen, stylesheets en scripts in afzonderlijke links worden opgehaald.

Daarom is `preload` misschien nog wel nuttiger dan `prefetch`. `preload` haalt belangrijke bronnen op voor de huidige pagina.

```
<link rel="preload" href="css/styles.css" as="style">
```

De voorgaande regel zorgt ervoor dat de webpagina pas wordt weergegeven als de stylesheet is geladen. Dat is het kenmerk van preload: de opbouw (*rendering*) en weergave van de pagina wordt uitgesteld tot alle preload-links binnen zijn.

Bij preload hoort het attribuut as. Daarmee wordt het mediatype aangegeven. Voorbeelden zijn style voor stylesheets, video voor videobestanden en font voor downloadbare lettertypen (zie hoofdstuk 5). De volgende regel zorgt voor het vooraf laden van een lettertype:

```
<link rel="preload" href="/fonts/lettertype.woff2" as="font" type="font/woff2">
```

Dan is er prerender. Hiermee wordt een webpagina vooraf geladen én op de achtergrond opgebouwd (gerenderd) en in het geheugen gezet. Zo is de pagina in een flits beschikbaar als die wordt opgevraagd. Let op het woord als. Voor prerender geldt hetzelfde als voor prefetch: voorspellen is moeilijk, vooral als het de toekomst betreft (Wim Kan). Het kan zijn dat bandbreedte wordt verstuurd aan een pagina die nooit wordt geladen.

Tot slot is er dns-prefetch. Hiermee worden geen gegevens geladen, maar wordt alleen op de achtergrond de domeinnaam vertaald naar het bijbehorende IP-adres (*DNS resolving*). Deze informatie komt van servers met grote tabellen met namen en nummers en het opzoeken ervan kan soms (veel) vertraging geven. Dan is het niet de pagina die op zich laat wachten, maar de informatie van de server(s) met de DNS-informatie (DNS staat voor Domain Name System). Door nu terwijl de bezoeker een pagina laadt ook alvast de nummers op te zoeken van andere domeinen waarin die pagina naar wordt gelinkt, kunnen die bronnen sneller worden geladen als de bezoeker ernaar vraagt. Vraagt die er niet om, dan worden er ook geen gegevens gedownload.

Nu halen moderne browsers al automatisch alle DNS-adressen op van domeinen die ze tijdens het laden van een HTML-pagina tegenkomen. Domeinen op volgende pagina's ziet de browser uiteraard niet. Voor die domeinen plaatst u <link>-elementen in de eerste pagina. Doe dat als volgt voor zoveel domeinen als u nuttig vindt:

```
<link rel="dns-prefetch" href="http://www.een_extra_domein.com">
<link rel="dns-prefetch" href="http://www.nog_een_extra_domein.com">
```

Meer informatie over preloading staat op developer.mozilla.org/en-US/docs/Web/HTML/Preloading_content.

Informatie over de overige waarden van rel is te vinden op developer.mozilla.org/en-US/docs/Web/HTML/Link_types.

Knoppen en hyperlinks

Volgens de HTML-specificatie is er een duidelijk verschil tussen links en knoppen, maar op het moderne web is het uiterlijke verschil lang niet altijd duidelijk. Bekijk als voorbeeld de afbeelding, gemaakt op www.smashingmagazine.com.



Afbeelding 4.10 Wie is de knop?

Beantwoord de volgende vragen om te beoordelen of een component een link of een knop moet zijn:

- Opent de component een andere locatie of pagina? Dan is het een link `<a>`.
- In alle andere gevallen is het een knop `<button>`.

In de afbeelding is het ding links een knop. Het is een verzendknop (`<button type="submit">`) die hoort bij het invoerveld voor het e-mailadres. Samen vormen ze een formulier (zie hoofdstuk 7). Het ding rechts is een link `<a>`. Klik erop en er wordt een pagina geopend met informatie over de conferentie. Uiterlijk zijn het twee knoppen, maar in code zijn ze dat duidelijk niet. Dat is zeker voor een spraakbrowser een belangrijk verschil, want die kan nu de bezoeker vertellen of die een actie gaat uitvoeren met een knop of naar een andere pagina navigeert met een link. Voor de webbouwer is het verschil ook belangrijk, want een button heeft unieke en handige eigenschappen die een link niet heeft.

Samenvatting

Dit hoofdstuk staat in het teken van de kern van internet: koppelingen oftewel: hyperlinks.

- Een hyperlink wordt gemaakt met het element `<a>`.
- Hyperlinks kunnen verwijzen naar externe documenten of afbeeldingen, maar ook naar inhoud op de eigen webserver.
- Een hyperlink kan alle elementen omvatten, behalve een andere hyperlink of een element `<button>`.
- Met een bladwijzer wordt naar een bepaald punt in dezelfde of een andere pagina gesprongen.
- Goede hyperlinks vragen wat denkwerk. Er zijn richtlijnen die helpen bij het maken van betekenisvolle hyperlinks.
- Met het veelzijdige element `<link>` worden externe bestanden gekoppeld: waaronder stylesheets en favicons.
- Het laden en weergeven van de pagina kan worden versneld met het attribuut `rel` en de waarden `preload`, `prefetch` en `prerender`.
- Er zijn belangrijke technische verschillen tussen hyperlinks en knoppen, hoewel ze er hetzelfde uit kunnen zien. Als een component een andere locatie of pagina opent, is het een link `<a>`.

In het volgende hoofdstuk gaan we het hebben over onder meer afbeeldingen en geschikte bestandsformaten, video, audio en andere inhoud die wordt opgehaald uit externe bronnen: *embedded content*.

Oefeningen

- Maak met de kennis uit dit en de voorafgaande hoofdstukken een pagina met vooral hyperlinks (startpagina), bijvoorbeeld een overzicht van sites die uw interesse hebben (bedenk met welke elementen u zo'n lijst van hyperlinks markeert).
 - Wilt u een fotogalerij maken, lees dan eerst hoofdstuk 5.
- Beoordeel uw website – of andere websites – kritisch op de richtlijnen voor gebruikersvriendelijke koppelingen die in dit hoofdstuk worden beschreven. Zoek een slecht voorbeeld op en probeer die pagina (in grote lijnen) opnieuw te maken met goede, betekenisvolle koppelingen.

Beeld, geluid en andere externe inhoud

Dit hoofdstuk gaat over inhoud die u niet rechtstreeks in de HTML-code van de webpagina zet. In plaats daarvan zorgt die code ervoor dat de inhoud wordt opgehaald en in de browser wordt getoond. Dat kunnen afbeeldingen, video en audio zijn, maar ook (delen van) andere websites. Naast uitleg over de elementen die daarbij worden gebruikt, wordt ook de nodige achtergrondinformatie gegeven over soorten afbeeldingen en video- en audiobestanden. Omdat deze inhoud altijd ergens moet worden opgehaald, wordt eerst ingegaan op URL's en de mapstructuur van een website.

U leert in dit hoofdstuk:

Wat een URL is en hoe de mapstructuur van een website is opgebouwd.

Welke typen afbeeldingen geschikt zijn en welke kenmerken ze hebben.

Het element toepassen.

Flexibel afbeeldingen gebruiken met <picture>.

Externe inhoud opnemen met <iframe>, <embed> en <object>.

Video en audio afspelen in een webpagina.

Inleiding

Wanneer u een afbeelding op een webpagina toont, of een filmpje of een geluidsfragment, zet u niet het bestand zelf in de HTML-code. U maakt alleen een verwijzing naar het bestand. Dat kan op dezelfde webserver staan als uw pagina, maar dat hoeft niet. Een filmpje bijvoorbeeld kan van de servers van YouTube komen en een liedje kan door Spotify worden gestreamd. In de HTML-code geeft u aan wat voor soort inhoud moet worden opgehaald en waar dat bestand is te vinden. De browser zorgt ervoor dat u het kunt zien of horen.

In HTML-termen gaat het hier om *embedded content*, oftewel ingesloten inhoud. Kenmerkend is dat een andere bron in de pagina wordt geïmporteerd en door de browser wordt weergegeven.

Rondom de uitleg van de elementen voor embedded content wordt ook informatie over de inhoud zelf gegeven: achtergrondinformatie over afbeeldingen, videocodecs, ondertitels, bestandsformaten voor geluid en dergelijke. Bovendien is het handig als u weet hoe u bestanden en mappen op uw site organiseert en op welke wijze u naar bestanden verwijst.

URL's en structuur van de website

Embedded content komt 'ergens' vandaan, er is een externe bron. Om de content te kunnen laden, moet er dus een verwijzing, een adres zijn. Is de bron een andere website, dan moet u een volledige URL (*uniform resource locator*, ook wel *absolute verwijzing* genoemd) gebruiken, bijvoorbeeld

```
https://www.youtube.com/watch?v=wIMu_Mui2f0  
https://www.instagram.com/p/BvKicwB11B
```

Staat de content op uw eigen website, dan kunt u een *relatieve verwijzing* maken. Zo'n verwijzing bevat niet het protocoldeel https:// en begint niet bij de bovenste map, maar gaat uit van de locatie (de map) van het bestand waarin de verwijzing staat.

Omdat niet altijd bekend is of een site via http:// of https:// wordt aangeboden, is het toegestaan om in een absolute URL de protocolnaam weg te laten. De verwijzing wordt dan:

```
//www.youtube.com/watch?v=wIMu_Mui2f0
```



Insluitcode

Sites zoals YouTube of Instagram verstrekken vaak een *embed code*. Met die code kunt u de content insluiten in bijvoorbeeld een compleet videovenster of een diavoorstelling. Dit wordt verderop uitgelegd.



Beveiligde verzending met https://

Tot voor kort was `http://` het standaardprotocol, maar sinds Google pagina's met een beveiligde verbinding voorrang geeft in de zoekresultaten, worden (gelukkig) steeds meer sites via `https://` verstuurd.

Hoe en waar uw website ook wordt gehost en hoeveel sites er ook op de servers van de hostingprovider staan, uw eigen sitestructuur is altijd een afgesloten, zelfstandig geheel waarvan de hoofdmap, de *root*, uw domeinnaam is. (Achter de schermen is er nog wel een andere mapstructuur, maar die ziet u gewoonlijk niet en u doet er ook niets mee.) In de hoofdmap staan uw HTML-bestanden, waarvan de hoofdpagina altijd `index.html` heet (of `index.htm`). Dat is de pagina die automatisch wordt geopend als u de domeinnaam in de browser typt. De hoofdmap kan submappen bevatten. Het is goed gebruik afzonderlijke mappen te maken voor onder meer afbeeldingen, stylesheets en scripts. Schematisch kan uw site er dan als volgt uitzien:

```
root (www.mijnsite.nl)
/css
  stijlen.css
/images
/foto
  foto1.jpg
  foto2.jpg
/icons
  icon1.png
  icon2.png
/scripts
  script.js
about.htm
index.htm
```

Typt u `www.mijnsite.nl` in de browser, dan wordt automatisch `index.htm` geopend. Dit bestand moet in de root van de site staan. (Als het ontbreekt krijgt u een 404-pagina: bestand niet gevonden.) Om vanuit `index.htm` met HTML-code een foto te laden, maakt u als volgt een relatieve verwijzing:

```
images/foto/foto1.jpg
```

Op deze manier dirigeert u de browser in de structuur van `//www.mijn-site.nl` naar de map `images` en naar de map `foto` om daar het bestand `foto1.jpg` op te halen. (De volledige (absolute) URL is dan: `//www.mijn-site.nl/images/foto/foto1.jpg`.) Hierbij wordt steeds een stapje dieper in de structuur van de website gezet.

Behalve omlaag kunt u ook omhoog in de structuur. Stel dat u vanuit een stylesheet in de map `css` een pictogram wilt laden, dan moet u vanuit die map een stapje omhoog naar de root en dan omlaag naar de map `images/icons`:

```
../images/icons/icon1.png
```

De twee puntjes geven aan dat eerst naar de bovenliggende map wordt gegaan, en daarna twee keer omlaag. Zou u twee mappen omhoog moeten, dan typt u de constructie `.../...` en voor drie mappen `.../.../...`.

Lokale site

Als u HTML-bestanden opent die u op uw eigen computer staan, is het verhaal over de mapstructuur precies hetzelfde. Het verschil is dat u in de adresbalk van de browser ziet `file:///C:/mapnaam/mapnaam/_/bestandsnaam`. Het protocol `file://` geeft aan dat het om het lokale bestandssysteem gaat en de derde `/` hoort bij de schijf waarop uw bestanden staan, hier `/C`. Hoe dit adres er precies uitziet hangt volledig af van de locatie van uw HTML-bestanden en uw eigen mapstructuur. Het kan handig zijn om uw eigen werk onder te brengen in een submap van Documenten, bijvoorbeeld `Documenten/www/handboek_html_css`. Dit adres is dan uw *siteroot*. Daarin plaatst u de HTML-bestanden en u maakt mappen voor de diverse bestandstypen, net als op een webserver. Als dit een overzichtelijke structuur is, kunt u de bestanden en mappen vanuit de lokale root één op één uploaden naar de webserver.



Gebruik nooit zelf `file://`

Gebruik tijdens het coderen *nooit* een verwijzing met `file://`. Als u een pagina met zulke verwijzingen op de webserver zet, zullen de bestanden waarnaar wordt verwezen nooit worden gevonden. Gebruik relatieve verwijzingen of absolute URL's.



Hoofdlettergevoelig

Webservers maken meestal onderscheid tussen hoofdletters en kleine letters in map- en bestandsnamen. Dat is iets om rekening mee te houden als bijvoorbeeld een map of bestand niet wordt gevonden wanneer u de site opent. Het is goed gebruik map- en bestandsnamen met kleine letters te schrijven.

Soorten afbeeldingen

Met CSS kan tekst fraai worden opgemaakt, maar desondanks is een webpagina zonder afbeeldingen bijna ondenkbaar. Al is het maar een logo. Op een webpagina kunt u diverse, maar niet alle typen afbeeldingen gebruiken. Globaal zijn er twee mogelijkheden: bitmaps en vectorafbeeldingen.

Bitmaps

Een bitmap of rasterafbeelding is opgebouwd uit pixels, beeldpunten. Het aantal pixels bepaalt hoe groot een afbeelding kan worden getoond, of beter: tot welke grootte de afbeelding er nog mooi uitziet. Een kleine bitmap die u groot laat weergeven krijgt kartelranden en ziet er niet uit. Dat komt doordat de pixels moeten worden vergroot om de gewenste ruimte op te vullen. Het is daarom belangrijk dat een foto voldoende pixels heeft voor de gewenste weergave. Op een beeldscherm is dat in het algemeen eenvoudig te bepalen. In de eigenschappen van de afbeelding ziet u hoeveel pixels er in de hoogte en breedte beschikbaar zijn en met die afmetingen kan de foto zonder problemen worden getoond. Het wordt een ander verhaal als u de gebruiker de mogelijkheid wilt bieden om het beeld in hoge kwaliteit af te drukken, want dan moet u ook rekening houden met de afdrukresolutie.

Bij foto's uit een digitale camera speelt overigens meestal het omgekeerde probleem: er zijn te veel pixels. Op zichzelf is dat niet erg, maar al die beeldinformatie moet ook over de dataverbinding worden getransporteerd en dat kan ervoor zorgen dat het lang duurt voordat de foto is geladen. In die situatie zult u daarom het beeld vooraf verkleinen. Een beproefd concept is een galerij van miniaturen die linken naar de afbeeldingen op groot formaat. De grote afbeelding wordt pas geladen als de gebruiker hem ook daadwerkelijk wil bekijken en op de link klikt.

Met de komst van hogeresolutieschermen op mobiele apparaten is het verhaal nog wel iets lastiger geworden, omdat dergelijke schermen in staat zijn haarscherpe foto's weer te geven. Dan moet zo'n foto ook wel in hoge resolutie beschikbaar zijn. Voor een bezoeker met een 'normaal' scherm zijn al die pixels echter alleen maar ballast. HTML biedt met het element `<picture>` in combinatie met media queries mogelijkheden om dit probleem (deels) te ondervangen; dit komen verderop aan de orde.

De kleuren van een bitmap zijn opgeslagen in de pixels. Een beeldscherm geeft kleur weer volgens het RGB-model. RGB staat voor de kleuren rood, groen en blauw. Elke pixel kan voor elke kleur een waarde hebben van 0 tot 255. Een rode pixel heeft de waarden 255, 0, 0, een groene pixel 0, 255, 0 en een

blauwe pixel 0, 0, 255. Hoe hoger de waarde, hoe groter de helderheid van de kleur. Zwart is 0, 0, 0 en wit is 255, 255, 255. Grijstinten ontstaan als de RGB-waarden gelijk zijn. Een middengrijs kleur is in RGB 128, 128, 128.

Niet elk type bitmapbestand is geschikt voor het web. Niet geschikt zijn bijvoorbeeld BMP, TIF, CDR enzovoort. Deze worden verder ook niet genoemd. De volgende typen zijn geschikt en worden verderop besproken:

- GIF
- JPG
- PNG
- WebP



Kleur in CSS

Op zichzelf hebben kleur en kleurmodellen niets met HTML te maken, maar des te meer met CSS. Beschouw deze uitleg als een beknopte inleiding. In de hoofdstukken over CSS wordt meer verteld over kleur.

Vectorafbeeldingen: SVG

Een vectorafbeelding is niet opgebouwd uit pixels. Het beeld wordt door wiskundige berekeningen samengesteld uit punten, lijnen, krommen en dergelijke. Een vectorafbeelding is een wiskundige beschrijving van de vormen en kleuren.

Het grote voordeel van vectorafbeeldingen boven bitmaps is dat ze niet lelijk worden als ze worden geschaald. Ze kunnen op elke grootte worden weergegeven. Dat is een reden om bijvoorbeeld logo's of pictogrammen als vectorafbeelding te maken.

Voor het tekenen van vectorafbeeldingen is speciale software nodig, bijvoorbeeld Adobe Illustrator of CorelDRAW. Een foto kunt u niet 'opslaan als' vectorafbeelding en gebruiken als schaalbaar beeld in een webpagina.

Vectorafbeeldingen hebben een eigen webstandaard: SVG. Dit staat voor *scalable vector graphics*. SVG is een markeertaal zoals HTML, maar dan een XML-dialect voor het beschrijven van afbeeldingen. Uitleg van deze taal voert te ver voor dit boek, maar de volgende SVG-code maakt een rechthoek:

```
<svg>
  <rect width="200" height="100" fill="#BB042A" />
</svg>
```

Deze rechthoek (*rectangle*) heeft een breedte van 200 pixels, een hoogte van 100 pixels en een olijfkleurige vulkleur.

Het aardige is dat u ook zonder kennis van de taal aan de slag kunt. Bijvoorbeeld Adobe Illustrator biedt de mogelijkheid om vectorafbeeldingen op te slaan als SVG-bestand. Om u een idee te geven, de volgende code is het resultaat van het opslaan als SVG-bestand van een rode cirkel gemaakt in Illustrator:

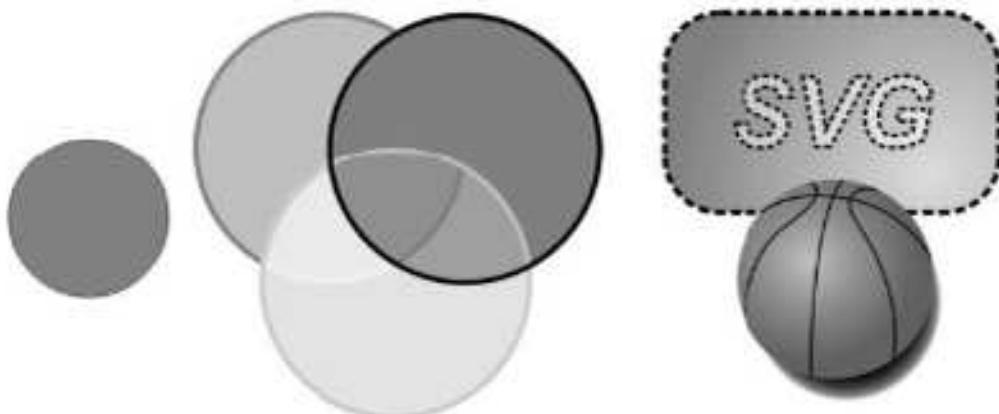
```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 201.3 204">
  <circle cx="100.8" cy="101.9" r="95.1" style="fill:#F00;stroke:#000"/>
</svg>
```

Tussen alle elementen en attributen herkent u het element `<circle>` met attributen voor de x- en y-coördinaten van het middelpunt van het tekengebied (`viewBox`) en de straal (`r`). Verder bevat `style` een vulkleur en een randkleur. Naar deze afbeelding (de naam is `cirkel.svg`) wordt in het HTML-document net zo verwezen als naar een bitmap:

```

```

De afbeelding kan ook als code in de HTML worden opgenomen. Plak gewoon het complete element `<svg>` in de HTML-code.



Afbeelding 5.1 Deze afbeelding ziet er op elke grootte scherp uit, probeer het zelf: upload.wikimedia.org/wikipedia/commons/1/15/Svg.svg.

Werken met SVG vergt extra studie, maar het is zeker de moeite waard als u het belangrijk vindt om schaalbare afbeeldingen te gaan gebruiken. Denk daarbij ook aan het groeiende aandeel van mobiele browser en verschillende beeldschermresoluties. SVG is bijvoorbeeld zeer geschikt voor interface-elementen (pictogrammen) en achtergronden. De specificatie SVG staat op www.w3.org/TR/SVG/. Een interessante bron is ook css-tricks.com/using-svg/.

Bestandsformaten voor bitmaps

Terug naar de ‘eenvoudige’ afbeeldingen: bitmaps. Voor webpagina’s zijn er de volgende bestandsformaten:

- WebP
- PNG, Portable Network Graphics
- JPEG, Joint Photographic Expert Group
- GIF, Graphics Interchange Format

We nemen ze in historische volgorde door.



Let op de rechten

Op foto’s en andere afbeeldingen die u willekeurig van internet downloadt, rust meestal copyright. Dat wil zeggen dat u de foto niet zonder toestemming van de maker mag gebruiken. Meestal wordt een vergoeding gevraagd. Er zijn ook allerlei websites waar u rechtenvrije foto’s kunt vinden. Twee voorbeelden zijn unsplash.com en pixabay.com, maar googelen naar free images levert nog veel meer sites op.

GIF

GIF en JPEG zijn de oudste bestandformaten voor het web. GIF heeft een kleurdiepte van 8 bits voor de drie standaardkleuren rood, groen en blauw. Een GIF-plaatje kan daardoor maximaal 256 kleuren bevatten. Dat is te weinig voor foto-kwaliteit, maar voldoende voor cartoons, logo’s en getekende afbeeldingen.

In GIF is naast de beperkte kleurdiepte ook een compressiemechanisme ingebouwd, waardoor de uiteindelijke bestanden nog maar een fractie van de omvang van het originele bestand hebben. Het compressiemechanisme LZW



Afbeelding 5.2 Een GIF-afbeelding geplaatst op een grijze ondergrond met transparante achtergrond (links) en zonder een transparante achtergrond.

dat wordt gebruikt komt vooral goed tot zijn recht als de afbeelding grote stukken dezelfde kleur bevat. Dit komt vaak voor bij logo's en andere plaatjes die met de computer worden gemaakt. Dit type compressie geeft geen kwaliteitsverlies. Dit comprimeren en decomprimeren gaat overigens razendsnel en volledig automatisch; de GIF-engine van de browser neemt dit voor zijn rekening.

Samenvattend zijn de kenmerken van GIF-afbeeldingen:

- maximaal 256 kleuren;
- lossless compressie (er gaat geen informatie verloren);
- animatie mogelijk;
- transparante achtergrond mogelijk;
- interlacing mogelijk (plaatje wordt lijn voor lijn opgebouwd, waardoor de ruwe afbeelding snel zichtbaar is);
- bruikbaar als afbeelding met hyperlink;
- aanbevolen voor cartoons, logo's en computertekeningen, afgeraden voor fotomateriaal.

JPEG

JPEG is het andere oertype voor afbeeldingen op websites. Daarbuiten wordt het op grote schaal gebruikt voor foto's uit digitale camera's. JPEG heeft een eigen compressiealgoritme ingebouwd en kan veel meer kleuren bevatten dan GIF. Omdat met 24-bits kleurdiepte wordt gewerkt, kunnen JPEG-afbeeldingen uit meer dan 16,7 miljoen kleuren bestaan. De compressiemethode is ideaal voor foto's en afbeeldingen met veel natuurlijke kleuren.

Afbeeldingen die met deze compressiemethode zijn gecomprimeerd, verliezen wel informatie. Dat wil zeggen dat het gedecomprimeerde bestand niet exact gelijk is aan het origineel. JPEG verwijdert informatie uit de afbeelding. Het compressiealgoritme gaat ervan uit dat het oog niet zo veel kleuren kan onderscheiden als in het origineel aanwezig zijn. De 'overbodige' kleuren worden gefilterd om ruimte te sparen. Deze verloren kleuren worden door het decompressiealgoritme niet teruggevonden.

Veel programma's die afbeeldingen in het type JPEG kunnen opslaan, laten de gebruiker de verhouding tussen compressiefactor en informatieverlies zelf bepalen. Een hogere compressiefactor levert zodoende een kleiner bestand op, maar met minder kwaliteit. Een lagere compressiefactor behoudt de kwaliteit, maar levert een groter bestand op.

U ziet dit in de afbeelding. Moet u in de praktijk een keuze maken: waarden tussen de 40 en 60 procent compressie leveren meestal nog prima beelden op voor gebruik op het beeldscherm, terwijl de bestandsomvang fors afneemt.



Afbeelding 5.3 Bij verschillende compressiefactoren gaat de kwaliteit van het bestand achteruit; de bestandsomvang wordt echter wel veel kleiner.

Samenvattend de kenmerken van JPEG-afbeeldingen:

- maximaal 16,7 miljoen kleuren;
- lossy compressie (er gaat informatie verloren bij het comprimeren);
- geen animatie mogelijk;
- geen transparante achtergrond mogelijk;
- interlacing niet mogelijk;
- bruikbaar als afbeelding met hyperlinks;
- aanbevolen voor fotorealistisch materiaal.



JPEG lijkt op MP3

Het compressiemechanisme JPEG voor afbeeldingen is vergelijkbaar met MP3 (of WMA of AAC) voor audiobestanden. Net zoals bij MP3-compressie vrijwel onhoorbare frequenties uit het originele signaal worden gefilterd en de rest van de informatie wordt samengedrukt om ruimte winst te boeken, zo worden bij JPEG-filtering onzichtbare kleuren uitgefilterd om een kleiner bestand te realiseren.

PNG

Het bestandstype PNG (Portable Network Graphics en uitgesproken als 'ping') is ontwikkeld als alternatief voor GIF in de tijd dat er nog patenten op GIF rustten. Hoewel die intussen zijn verlopen, heeft PNG dankzij technische voordelen langzamerhand de rol van GIF overgenomen:

- ondersteuning van 48-bits kleurdiepte of een 16-bits grijswaardenschaal;
- alfakanalen (variabele transparantie tegenover statische transparantie van GIF);

- gammacorrectie (helderheidcorrectie en -controle over verschillende platforms, zodat een afbeelding er op een Macintosh even helder uitziet als op een Windows-systeem);
- tweedimensionale interlacing (progressieve opbouw van de afbeelding tegenover eendimensionale interlacing bij GIF);
- globaal een 5 tot 25 procent betere compressie dan bij GIF (behalve bij kleine bestanden). De compressiemethode is evenals bij GIF lossless, wat wil zeggen dat er geen informatie wordt weggelegd uit de afbeelding (zoals bij JPEG).

Er is ook een eigenschap die PNG explicet niet ondersteunt: animatie. Er wordt gewerkt aan twee PNG-typen die wel animatiemogelijkheden bieden, maar deze APNG- en MNG-formaten worden nog amper ondersteund. Alleen Firefox en Safari kunnen ermee overweg. APNG en MNG lijken een eeuwige belofte te blijven, want vooruitgang is de laatste jaren nauwelijks te bespeuren.

WebP

WebP (uitgesproken als 'weppie') is het nieuwste bestandtype voor afbeeldingen op het web. Het is een aankoop van Google, dat het nu verder ontwikkeld. WebP ondersteunt zowel lossless als lossy compressie. Google claimt dat de WebP-afbeeldingen zonder verlies 26% kleiner zijn dan PNG's en dat WebP-afbeeldingen met verlies 25-34% kleiner zijn dan vergelijkbare JPEG's. WebP ondersteunt transparantie in afbeeldingen met en zonder verlies. Daarnaast ondersteunt WebP animatie.

Moderne browsers kunnen WebP weergeven, maar toch is er wel een probleempje: het converteren naar het WebP-formaat. Voor Photoshop is er een plug-in (telegraphics.com.au/sw/product/WebPFormat#webpformat) en in Sketch is het ingebouwd (www.sketchapp.com). Op developers.google.com/speed/webp zijn instructies te vinden voor andere methoden om afbeeldingen te converteren naar WebP.

De juiste afbeelding voor het scherm

'Vroegah' was het gebruik van afbeeldingen makkelijk. In 2009 had 60% van de computergebruikers een beeldscherm met een resolutie die varieerde van 1024 x 768 pixels tot 1280 x 1024. Voor een weergave op volledige grootte van een foto op een monitor van 17 inch was een afbeelding van 1280 bij 1024 pixels genoeg. Vandaag is de variatie in afmetingen en resoluties van weergaveapparaten enorm. Dat is fijn voor ons als consument, maar het is wel wat lastiger voor ons als webbouwers. Die foto in hoge resolutie voor dat 4K-beeldscherm van 27 inch wil een smartphonegebruiker niet hoeven downloaden.

Andersom doet u de iPad-gebruiker geen plezier met de postzegel die voldoende is op een eenvoudige telefoon.

Dan is er nog het verschil in ruimte. Een grote foto wordt op een telefoonscherm zo verkleind dat er van het beeld niet veel overblijft. Het zou beter zijn als er voor kleinere schermen een uitsnede van de grote foto kon worden geladen.

De oplossing van al deze tegenstrijdige belangen zit in twee dingen:

- afbeeldingen in verschillende formaten;
- een mechanisme dat ervoor zorgt dat de browser de meest geschikte afbeelding download.

Beide zijn beschikbaar. Tenminste, als u zorgt voor de geschikte foto's. HTML en CSS beschikken over prima hulpmiddelen om de browsers het juiste beeld te laten gebruiken. Wat HTML betreft gaan we in op twee elementen: `` en `<picture>`. Hiervan is `` het belangrijkst, want dat zorgt in alle situaties voor de weergave. Dat element wordt eerst besproken voor algemeen gebruik en daarna wordt ingegaan op werken met responsieve afbeeldingen.



Veelzijdig en veelbesproken onderwerp

Het gebruik van de juiste afbeelding onder verschillende omstandigheden (*responsive images*) is een veelomvattend onderwerp dat we hier onmogelijk uitputtend kunnen behandelen. Het is echter ook een veelbesproken onderwerp waarover talloze artikelen zijn te vinden. Google maar eens 'responsive images'. Een aanrader is developer.mozilla.org/en-US/docs/Learn/HTML/Multimedia_and_embedding/Responsive_images. In hoofdstuk 10 wordt nog ingegaan op de viewport en de device pixel ratio, onderwerpen die hiermee samenhangen.

Afbeeldingen plaatsen met ``

`` is het element dat ervoor zorgt dat in een HTML-pagina een afbeelding te zien is. Het is voorgaande hoofdstukken al diverse keren langsgekomen, maar de mogelijkheden ervan zijn nog niet uitgelegd. In de eenvoudigste vorm ziet `` er zo uit:

```

```



Leeg element

`` is een leeg element. U mag de zelfsluitende tag `` gebruiken, maar dat moet niet.

Naast de globale attributen (waaronder `class` en `id`) kan dit element diverse attributen hebben waarmee bijvoorbeeld de bron, de breedte en de hoogte worden ingesteld. Er is ook een attribuut om een afbeelding van klikbare gebieden te voorzien die fungeren als hyperlink.

De bron: `src`

Eén attribuut is in elk geval onmisbaar: `src`. Dit attribuut geeft aan waar de afbeelding is te vinden. In het voorbeeld

```

```

wordt `afbeelding.jpg` opgehaald uit de map `images`. Deze verwijzing moet compleet zijn. Dat is een open deur, maar in de paragraaf *URL's en structuur van de website* leest u hoe u die deur op de juiste manier opent.

Beschrijvende tekst: `alt`

Het attribuut `alt` verschaft een alternatieve tekst voor situaties waarin de afbeelding niet zichtbaar is. Het is het terugvalmechanisme van `src`. Onderschat niet het belang hiervan. Bezoekers met (visuele) beperkingen kunnen een browser gebruiken die geen beeld toont of de pagina voorleest, maar ook kan het weergeven van afbeeldingen in de browser zijn uitgeschakeld; of de afbeelding is domweg niet beschikbaar door een storing of foute verwijzing. In al die gevallen is wel de alt-tekst zichtbaar (of deze wordt voorgelezen). Dit maakt dat het zinvol is goed na te denken over de inhoud ervan. Stel u voor dat u het beeld niet ziet en vraag u af welke informatie u dan mist. Dat is de alt-tekst.

Als voorbeeld een fictief bericht over het uit de vaart nemen van het fietsveer over de rivier de Lek bij Bochtig. Fietser moeten daardoor straks kilometers omrijden. Het beeld bij dit artikel is als volgt gecodeerd:

```
<figure>

<figcaption>Fietser onderweg naar de veerstoep bij Bochtig.</figcaption>
</figure>
```

Merk op dat in dit voorbeeld de alt-tekst het beeld zo goed mogelijk beschrijft en dat het bijschrift veel beknopter is. (Zie hoofdstuk 4 voor uitleg over `<figure>` en `<figcaption>`.)

Voorbeelden van alt-gebruik

Er zijn situaties waarin de alt-tekst leeg kan zijn. Een lege alt-tekst geeft aan dat de afbeelding niet belangrijk of alleen versiering is en dat een spraakbrowsert geen beschrijving hoeft te geven. U codeert dat als lege tekenreeks:

```

```

Enkele situaties met daarbij de verwachte alt-tekst:

- grafische knop zonder tekst: de functie van de knop;
- logo zonder tekst: de naam van de organisatie;
- logo met HTML-tekst los erbij: lege alt-tekst;
- pictogram zonder tekst: de functie van het pictogram;
- pictogram met HTML-tekst: lege alt-tekst;
- inline afbeelding (smiley): de uitdrukking van de smiley;
- puur decoratief beeld: lege alt-tekst.



Meer voorbeelden

Bij de specificatie worden veel duidelijke voorbeelden gegeven. Kijk op www.w3.org/TR/html/semantics-embedded-content.html#alt-text.

Afmetingen: width en height

De afmetingen van een afbeelding worden aangeduid met width en height.

De belangrijkste reden om deze attributen te gebruiken is dat de browser dan weet wat er komt. De aangegeven breedte en hoogte worden bij het opbouwen van de webpagina in het browservenster gereserveerd. Ontbreken de attributen en is de afbeelding pas volledig geladen als de andere inhoud er al staat, dan moet al die inhoud oopschuiven; de pagina wordt opnieuw opgebouwd en het beeld verspringt daardoor. Dat geeft geen goede gebruikerservaring.

Het gebruik van deze attributen is eenvoudig:

```

```

De eenheid van de waarden is in pixels en niets anders. Daarom hoeft u de eenheid ook niet aan te geven. Gebruik wel altijd beide attributen.

Het is geen goed idee om de attributen te gebruiken om afbeeldingen te schalen. Een digitale foto van 4000 bij 3000 pixels van een kleine 8 MB kunt u natuurlijk een width en height geven van 400 en 300, maar dan wordt nog steeds de volle 8 MB van het originele beeld gedownload. Bovendien loopt u

het risico dat u verkeerde waarden gebruikt als de getallen niet zo mooi zijn als in dit voorbeeld, waardoor het beeld wordt vervormd. Gebruik daarom altijd een beeldbewerkingsprogramma om de afbeeldingen de gewenste afmetingen in pixels te geven.

Voor afbeeldingen die niet voor de gebruiker bestemd zijn, kunnen width en height de waarde 0 krijgen. Dat kan bijvoorbeeld een afbeelding zijn die bij een bezoekersteller hoort.

Klikbare gebieden: usemap

Wilt u op een kaart van Nederland de regio's waarin uw bedrijf opeert klikbaar maken met een hyperlink naar elke regiokantoor? Dan gebruikt u het attribuut usemap op de afbeelding van een kaart. Met usemap, de elementen <map> en <area> en attributen voor vormen en coördinaten markeert u gebieden in een afbeelding die elk een eigen hyperlink kunnen hebben. Het klinkt ingewikkelder dan het is. Ingewikkelde vormen waardoor veel coördinaten nodig zijn, kunnen de code echter lastig leesbaar maken. (En het vinden van de punten kan een lastige kus zijn.) Om niet te verdrinken in getallen wordt een eenvoudig voorbeeld gebruikt.

De basisafbeelding is een witte rechthoek met een rand en daarin drie vormen. Deze basisafbeelding is niet klikbaar. De vormen zijn dat wel en ze hebben elk een eigen hyperlink naar een ander bestand. De code is als volgt:

```


  <area shape="rect" coords="30,25,225,225" href="vierkant.htm">
  <area shape="circle" coords="360,120,100" href="cirkel.htm">
  <area shape="poly" coords="450,225,650,225,550,25" href="driehoek.htm">

```



Afbeelding 5.4 Klikbare gebieden met hyperlinks. Buiten die gebieden is er geen interactie (05_04.html).

- De eerste regel definieert de afbeelding en een verwijzing naar de usemap: #vormen.
- De tweede regel markeert met het element <map> de usemap met de naam vormen.
- De derde regel definieert een rechthoek (rect). De waarden 30,25 markeert het punt linksboven en 225,255 is het punt rechts onder.
- De vierde regel definieert een cirkel (circle). De waarde 360 markeert het middelpunt van de cirkel ten opzichte van de linkerkant van de afbeelding. De waarde 120 is het middelpunt ten opzichte van de bovenkant van de afbeelding. De derde waarde is de straal: 100.
- De vijfde regel markeert een veelhoek (poly). Deze vorm heeft minimaal zes coördinaten, twee voor elk hoekpunt, bestaande uit de x- en y-waarde ten opzichte van de hoek linksboven van de afbeelding. Deze driehoek is daarmee een minimale veelhoek en bij ingewikkelde vormen hebt u veel meer coördinaatparen.
- In de laatste regel wordt het element afgesloten.

Responsive images

Dan nu de magie die met kan worden bedreven: afbeeldingen laden gebaseerd op schermresolutie en schermgrootte. We bespreken drie scenario's:

- schermen met hoge resolutie;
- afbeeldingen met variabele afmetingen;
- bijgesneden of veranderde afbeeldingen.

Bedenk voor u aan de slag gaat dat niet alle afbeeldingen de extra behandelingen nodig hebben. Het betreft sowieso alleen bitmaps, want vectorafbeeldingen zijn vanzelf al schaalbaar. Daar hoeft u geen verschillende versies van te maken. Als er weinig verschil is in bestandsgrootte tussen de grote en kleine afbeeldingen kunt u zich ook de moeite besparen. Tot slot, mocht het paginaontwerp niet flexibel zijn, dan volstaat ook één afbeelding in een vaste grootte, eventueel wel met verschillende versie voor verschillende resoluties.

Srcset voor schermen met hoge resolutie

Het haarscherpe beeld op bepaalde smartphones komt (onder meer) doordat die schermen naar verhouding enorm veel pixels hebben. De resolutie 1080 x 1920 (full HD) op een scherm van 5,5 inch is geen uitzondering. Als dat dezelfde pixels zouden zijn als op een desktopmonitor, was de inhoud onleesbaar klein. De details rond fysieke pixels, CSS-pixels en device pixel ratio leest u in hoofdstuk 10, maar al die pixels worden door de browser omgerekend naar een beeld dat overeenkomt met wat u op een standaarddesktopmonitor zou

zien. De omrekenfactor is de *device pixel ratio* en die is bij een smartphone met een scherm van circa 5,5 inch en een resolutie van 1080 x 1920 meestal 3. Deel de resolutie door de device pixel ratio en u hebt de zichtbare resolutie, hier 360 x 640 pixels (de CSS-pixels).

Voor ‘gewone’ HTML-elementen is dit dus een scherm van 360 x 640. Maar afbeeldingen kunnen bij datzelfde formaat een drie keer hogere resolutie hebben: de genoemde 1080 x 1920. Zo’n afbeelding is groter en kost daardoor meer dataverkeer, maar geeft een haarscherp beeld. Het is alleen niet voor elke gebruiker zinvol om zo’n grote afbeelding te downloaden. Dan komt het attribuut `srcset` in beeld.

Met `srcset` worden verschillende afbeeldingen aangeboden waaruit de browser de meest geschikte kiest op basis van de waarde bij `x`. Hierbij staat `x` voor de device pixel ratio van het weergaveapparaat. De waarden worden gescheiden door een komma. In het volgende voorbeeld kan de browser – die de device pixel ratio van het apparaat detecteert – kiezen uit drie afbeeldingen voor een device pixel ratio van 1, 2 of 3.

```

```

U ziet ook het attribuut `src` met een verwijzing naar een afbeelding. Dat is het terugvalmechanisme voor browsers die `srcset` niet ondersteunen.



Testen welke afbeelding wordt getoond

Welke afbeelding uit de `srcset` is geladen, kunt u snel testen door in het snelmenu Afbeelding opslaan als te kiezen. In het venster Opslaan als ziet u de bestandsnaam.

Srcset voor variabele afmetingen

Een modern ontwerp is altijd responsive en dan is een selectiemechanisme op basis van alleen device pixel ratio niet de beste keus. Dat brengt ons bij de volgende mogelijkheid van het attribuut `srcset`: selecteren op basis van breedte. Hierbij wordt `srcset` gecombineerd met het attribuut `sizes` en wordt gebruikgemaakt van media queries (zie ook hoofdstuk 10).

De kern van deze methode is dat afbeeldingen van verschillende grootte worden aangeboden aan de browser. In plaats van een `x` wordt nu een `w` met een

breedtewaarde toegevoegd. De waarde bij `w` is niet willekeurig. Het is de daadwerkelijke breedte waarmee u de afbeelding (bijvoorbeeld na bewerking in Photoshop) hebt opgeslagen. De bestandsnaam mag zijn wat u wilt, maar de `w`-waarde moet kloppen.

```

```

De browser heeft nu nog niet genoeg informatie voor het kiezen van de juiste afbeelding. Die levert het attribuut `sizes`. Dit attribuut bevat een of meer media queries (voorwaarden) voor de venstergrootte (preciezer: de `viewport`) gevuld door de gewenste grootte van de afbeelding. De laatste waarde is alleen een afbeeldingsgrootte. Het volgende voorbeeld betekent:

- Bij een vensterbreedte tot 640 pixels mag een afbeelding net zo breed zijn als het venster (100vw is 100% van de viewportbreedte, zie hoofdstuk 8, de eenheid `vw`).
- Bij een venster tot 1280 pixels mag de afbeelding half zo breed zijn als het venster (50vw).
- Bij een venster tot 1920 pixels mag de afbeelding een derde van het venster zijn (33vw).
- Bij venster dat niet aan de voorgaande voorwaarden voldoet (dus 1921px en meer, mag de afbeelding 100% van de vensterbreedte zijn (100vw).

In code:

```

```

Het complete element `` bevat nu een basisafbeelding in `src`, de keuzemogelijkheden in `srcset` en aanwijzingen voor gebruik bij bepaalde vensterafmetingen. Met deze informatie bepaalt de browser welke afbeelding het best

past bij de omstandigheden. U geeft dus alleen aanwijzingen, de browser maakt de keus.

Let op, als eenmaal een groter formaat afbeelding is geladen en in de browsercache is opgeslagen, wordt die afbeelding op elke schermbreedte gebruikt. Dat is iets om rekening mee te houden bij het testen. De ontwikkelaarshulpmiddelen hebben een optie om caching uit te schakelen. Bij Firefox wordt de cache HTTP-buffer genoemd.

Analyse van de code

- De code en de afbeeldingen zijn beschikbaar in 05_img.html.

Test dit voorbeeld in de browser. Welke afbeelding wordt met het codevoorbeeld bij de verschillende schermbreedten geladen? Spoiler: dit is een strikvraag.

De browser laadt bij elke schermbreedte tot 1920px het bestand waterlelie_640.jpg! Pas daarboven wordt de hires-foto geladen. Ga maar na hoe groot de afbeelding effectief moet zijn bij de verschillende sizes:

- 640px op 100vw = 640px
- 1280px op 50vw = 640px
- 1920px op 33vw = 640px
- 1921px en groter op 100vw = minimaal 1921px (waterlelie_hires.jpg)

De browser bekijkt dus per instelling welke afbeelding er daadwerkelijk nodig is voor een optimaal plaatje.



Beperk de breedte met max-width

Het komt nog aan de orde bij CSS, maar u kunt bij afbeeldingen waarvan het formaat kan variëren (responsive images dus) tegen onverwachte horizontale schuifbalken aanlopen. Dat gebeurt als de afbeelding breder is dan zijn container, bijvoorbeeld het venster of de viewport. De CSS-eigenschap `max-width: 100%` voorkomt dergelijke problemen. Met deze instelling kan de afbeelding wel smaller zijn dan het venster; het is een maximum, geen absolute waarde.

Bijgesneden afbeeldingen: <picture>

Soms geeft alleen een keuze uit verschillende afbeeldingformaten niet het gewenste resultaat. Denk aan een brede headerafbeelding. Als die terugschalt van 1200px naar 320px blijft er van het plaatje weinig over. Dat wordt zeker een probleem als het ook een beeldbepalend element bevat. In die situatie zou het handig zijn om een bijgesneden variant van de afbeelding te

kunnen laden. Ook daar bevat HTML een ingebouwd mechanisme voor: het element `<picture>`.

Het element `<picture>` lijkt een beetje op audio en video (zie hierna). Het is een container voor verschillende afbeeldingen, waarbij de browser kiest op basis van een of meer media queries. Het voorbeeld is gemaakt met een headerafbeelding in vier formaten, elk met een andere uitsnede. Drie ervan staan in een `<source>` met een media query en een bron in `srcset`. De vierde heeft alleen een `srcset` met de afbeelding die wordt getoond als niet aan de voor-gaande voorwaarden wordt voldaan. Dan volgt het verplichte element ``, zonder werkt het niet. Dat bevat de afbeelding die wordt getoond als de browser `<picture>` niet ondersteunt.

```
<picture>
  <source media="(max-width: 360px)" srcset="images/zeilschip-320.jpg">
  <source media="(max-width: 800px)" srcset="images/zeilschip-800.jpg">
  <source media="(max-width: 1024px)" srcset="images/zeilschip-1024.jpg">
  <source srcset="images/zeilschip-1920.jpg">
  
</picture>
```

- De code en de afbeeldingen zijn beschikbaar in 05_picture.html.

Het type afbeelding selecteren

`<picture>` en `<source>` kunnen worden gecombineerd met `type`. Dat geeft de mogelijkheid om de browser een afbeelding van een bepaald bestandsfor-maat te laten kiezen. Zo kan bijvoorbeeld het nog niet door elke browser ondersteunde bestandsformaat WebP worden gebruikt. Met `type` wordt het nieuwe bestandsformaat aangeboden, en de browser kan altijd terugvallen op een bekend bestandsformaat.

```
<picture>
  <source srcset="images/webp-voorbeeld.webp" type="image/webp">
  
</picture>
```

- Het voorbeeld en de afbeeldingen zijn beschikbaar in 05_webp.html.

Het verschil tussen `` en `<picture>`

Er lijkt misschien weinig verschil te zijn tussen `` met `srcset` aan de ene kant en `<picture>` met `<source>` aan de andere kant. Toch is dat er wel dege-lijk en het zit vooral in de mate van controle. Bij `` en `srcset` krijgt de

browser diverse afbeeldingen aangeboden. Op basis van device pixel ratio of viewportgrootte berekent de browser wat onder die omstandigheden de geschiktste afbeelding is en laadt die. Daar hebt u geen controle over. Bij `<picture>` en `<source>` hebt u juist veel controle over het resultaat. U geeft precies aan welke afbeelding bij een bepaalde venstergrootte hoort en als die voorwaarde klopt, wordt die afbeelding geladen. Verandert de viewport, dan wordt ook een andere afbeelding geladen. Het is dus een volledig responsieve oplossing. U kunt ze ook combineren; daarvan zijn in de specificatie ook voorbeelden te vinden, zie www.w3.org/TR/html/semantics-embedded-content.html#embedded-content-introduction.

Externe HTML-inhoud: `<iframe>`

Een webpagina kan links hebben naar andere pagina's en andere websites, maar het is niet mogelijk om een gewone hyperlink een andere pagina te openen *in* de huidige pagina. Toch is dat soms handig, bijvoorbeeld om extra informatie, advertenties of een twitterfeed te tonen.

Voor dergelijke situaties is er het element `<iframe>`. Dit markeert wat wordt genoemd een *nested browsing context*. Zonder in te gaan op de technische achtergrond wil dit zeggen dat in de aanroepende pagina in een afgebakend gebied een complete HTML-pagina wordt ingesloten die zijn eigen bronnen gebruikt en werkt als een pagina die in een nieuw venster zou zijn geopend. Nu valt een externe pagina buiten uw beheer en dat kan een veiligheidsrisico opleveren. De pagina zou bijvoorbeeld met scripting kunnen proberen de bezoeker malware of een virus in de maag te splitsen. Om dergelijke risico's uit te sluiten kan in `<iframe>` een veiligheidsmechanisme worden gebruikt.

Naast de globale attributen heeft `<iframe>` attributen voor onder meer de bron, beveiliging en de afmetingen.

Attributen van `<iframe>`

Het attribuut `src` bepaalt de bron van het `<iframe>`. Dit moet een volledige URL zijn:

```
<iframe src="http://www.domeinnaam.nl/pagina.htm"></iframe>
```



URL zonder http:// of https://

Merk op dat in de URL het protocol wordt weggelaten. Dit voorkomt problemen bij het laden als een pagina via http wordt opgevraagd, maar via https wordt verstuurd (en andersom). Veel https-websites leiden een aanvraag via http automatisch om naar https, maar dat gebeurt niet altijd.

In plaats van een document te laden kan ook de HTML-code in het `<iframe>` worden geschreven. Die code komt in het attribuut `srcdoc`. Houd er rekening mee dat u niet dezelfde aanhalingstekens kunt gebruiken voor het attribuut en de inhoud ervan. Voor een foutbestendige werking van HTML-code in `srcdoc` moet u de aanhalingstekens en eventuele ampersands 'escapen': u gebruikt de code `"` en de ampersand is `&`. U ziet dit in de volgende code:

```
<iframe srcdoc="<h1>Zonnebloem</h1>
<blockquote cite=&quot;//nl.wikipedia.org/wiki/Zonnebloem&quot;>
  <p>De zonnebloem (<i>Helianthus annuus</i>) is een tot 3 meter hoge, eenjarige plant uit de composietenfamilie (<i>Asteraceae</i>). De zonnebloem kan gezaaid worden van april tot half juni.</p>
</blockquote>
<img src=&quot;images/zonnebloem.jpg&quot; style=&quot;max-width:100%&quot;>">
</iframe>
```

Zonnebloem

De zonnebloem (*Helianthus annuus*) is een tot 3 meter hoge, eenjarige plant uit de composietenfamilie (*Asteraceae*). De zonnebloem kan gezaaid worden van april tot half juni.



Afbeelding 5.5 Een iframe. De inhoud is toegevoegd met het attribuut `srcdoc` (05_05.html).



Aanhalingsstekens weglaten

In HTML 5 zijn aanhalingsstekens bij attribuutwaarden niet verplicht (mits de waarde geen spaties bevat). U zou voor de leesbaarheid in dit voorbeeld de aanhalingsstekens weg kunnen laten. Dat is geen *best practice* voor normale HTML-code.

In `srcdoc` kan alle HTML-code worden geschreven die ook binnen `<body>` kan worden gebruikt. Worden `srcdoc` en `src` beide gebruikt, dan is `src` het terugvalmechanisme voor browsers die `srcdoc` niet ondersteunen. Met andere woorden: als de code in `srcdoc` niet wordt uitgevoerd, wordt de in `src` gespecificeerde webpagina geladen; in die volgorde.

Met de attributen `width` en `height` wordt de grootte van het `<iframe>` bepaald. Zonder deze attributen krijgt het element meestal een standaardformaat van 300 bij 150 pixels. (Dergelijke eigenschappen worden bij voorkeur in een stylesheet opgenomen.)



Schuifbalken

Gebruik voor het beïnvloeden van de schuifbalken de CSS-eigenschap `overflow`.

Beveiliging

Vanwege het potentiële gevaar van externe pagina's mag er in een `<iframe>` erg weinig:

- het document wordt behandeld als afkomstig van een unieke bron;
- het indienen van formulieren is uitgeschakeld;
- scriptuitvoering is geblokkeerd;
- API's zijn uitgeschakeld;
- links kunnen geen andere browsercontexten activeren;
- er kunnen geen plug-ins worden uitgevoerd met `<embed>` of `<object>`;
- de content kan niet naar de omsluitende pagina navigeren;
- automatisch geactiveerde functies zijn uitgeschakeld (bijvoorbeeld automatisch startend geluid).

Het attribuut `sandbox` is voor het beheer van die beperkingen. Een beperking wordt uitgeschakeld door de bijbehorende waarde op te nemen bij het attribuut `sandbox` in de tag `<iframe>`:

```
<iframe sandbox="" src="...>
```

Met onder meer deze waarden voor sandbox worden de bijbehorende beperkingen opgeheven:

- allow-forms: formulieren mogen worden ingestuurd;
- allow-pointer-lock: API's zijn ingeschakeld;
- allow-popups: pop-ups zijn toegestaan;
- allow-same-origin: de inhoud van het frame wordt behandeld alsof die van dezelfde bron afkomstig is;
- allow-top-navigation: staat de inhoud van het frame toe te navigeren naar de bovenliggend pagina.

Waarden worden gescheiden door een spatie. In het volgende voorbeeld zijn alle beperkingen van kracht, maar mag een formulier wel worden ingediend.

```
<iframe sandbox="allow-forms" src="http://www.andere-website.nl/pagina.htm">
```

Insluiten met <embed> of <object>

De elementen `<embed>` en `<object>` zijn bedoeld om externe inhoud anders dan HTML in te sluiten. (Want voor HTML-inhoud is er het `<iframe>`.) Ooit werden de elementen gebruikt om Flash-filmpjes of Silverlight-media in de pagina op te nemen, maar dergelijke technologieën komen amper nog voor op internet (in maart 2019 is Flash nog te vinden op 3,7 procent van alle websites en Silverlight op 0,1 procent). Voor webvideo en -audio zijn andere bestandsformaten beschikbaar gekomen en die worden geplaatst met de HTML 5-elementen `<video>` en `<audio>`. Zie hierna. Eigenlijk kunnen we geen toepassingen meer bedenken voor deze elementen, want zelfs voor het weergeven van een pdf-bestand hebben we genoeg aan het hyperlinkelement `<a>` (zie hoofdstuk 4, de paragraaf *Een pdf-bestand weergeven of downloaden*)!

Video in een webpagina

Het was allereerst webontwikkelaars en gebruikers en in navolging daarvan de ontwikkelaars van HTML een doorn in het oog dat het open web voor verrijkende toepassingen zoals video en audio afhankelijk was van beschermd technieken, zoals Flash, QuickTime, Real Player en Windows Media Player. In HTML 5 is dit probleem (grotendeels) opgelost met de introductie van de elementen `<video>` en `<audio>`. De ontwikkelingen zijn snel gegaan en alle moderne browsers ondersteunen de HTML 5-video en -audiomogelijkheden.

Alle browsers? Ja, maar met een kanttekening. Niet elke browser ondersteunt elk type video en zo vergt het toch nog wat extra werk om cross-browservideo in te bouwen, eventueel met een terugvalmechanisme voor antieke browsers.

Bestandstypen voor video

Als u online een filmpje bekijkt, heeft dat als extensie bijvoorbeeld avi, mov, mp4 of wmv. Die extensies staan niet voor het soort beeld en geluid dat u ziet, het zijn containerbestanden die video en audio bevatten die op een bepaalde manier zijn gecodeerd. Om de inhoud van zo'n bestand te kunnen lezen, moet op de computer de juiste decodeersoftware beschikbaar zijn: de codec.

Er zijn heel veel soorten codecs voor beeld en geluid in omloop, maar daarvan worden er drie het meest toegepast (tussen haakjes de staat de extensie):

- WebM (VP8 /VP9) De codecs VP8 en VP9, gebruikt in het videoformaat WebM, zijn kwalitatief hoogwaardig. In 2010 kocht Google het bedrijf dat de codec heeft ontwikkeld. Ook deze codecs zijn open source en worden ondersteund door Chrome, Firefox, Edge en Opera. Voor Internet Explorer en Safari (QuickTime) zijn er plug-ins. Deze codec heeft de voorkeur.
- H.264 (mp4) De codec H.264 voor video (en AAC voor audio) is geen open standaard. De codec geeft een goede beeldkwaliteit bij een hoge compressie, maar er rusten allerlei patenten op. Het kan in alle browsers worden afgespeeld, hoewel soms met een omweg via het besturingssysteem om de patenten te omzeilen.
- Ogg Theora (ogg) Ogg Theora (en Ogg Vorbis voor audio) is een open standaard met een zeer acceptabele kwaliteit. Deze codec wordt ondersteund door Firefox, Chrome en Opera.

Een codec kiezen

De HTML5-specificatie noemt geen codecs. Browserfabrikanten bepalen zelf welk formaat ze willen ondersteunen. Een paar jaar terug was het daardoor nog onvermijdelijk om verschillende videobestanden aan te bieden. Omdat nu alle browsers overweg kunnen met H.264 (mp4), zou u het daarbij kunnen laten. Wilt u meer zekerheid dat er geen bezoekers buiten de boot vallen, dan biedt u twee formaten aan om het hele spectrum af te dekken: mp4 en WebM. Dat is voor de code geen enkel probleem: het element `<video>` biedt de mogelijkheid om eenvoudig verschillende videobestanden aan te bieden, met verschillende codecs. Dan moeten die bestanden natuurlijk nog wel even worden gemaakt.

Video coderen

Voor het coderen van video zijn veel mogelijkheden beschikbaar, van uitgebreide (dure) pakketten tot eenvoudiger gratis software, al dan niet online werkend. Beschikt u niet al over videosoftware en wilt u experimenteren, dan kunt u kijken naar Freemake Video Converter (www.freemake.com), HandBrake voor mp4 (open source, www.handbrake.fr) en FFmpeg voor WebM (open source, www.ffmpeg.org).

Het element <video>

Het element <video> kan behalve voor video ook worden gebruikt voor audiobestanden met ondertitels. Met alleen het element in de pagina gebeurt er niets. De attributen zorgen ervoor dat een bestand wordt geladen en dat de diverse opties worden ingesteld.

- **src** Dit attribuut bevat de URL naar het videobestand. Gewoonlijk gebruikt u src niet in het element <video>, maar in het element <source> (zie hierna).
- **width** en **height** U geeft de afmetingen van de speler aan met width en height. De afmetingen kunnen uiteraard ook met CSS worden ingesteld.
- **controls** Dit attribuut zorgt ervoor dat de bedieningsknoppen worden getoond. Zonder dit attribuut zijn die onzichtbaar. Dit attribuut heeft geen waarde (het is een booleaans attribuut), alleen controls is voldoende.
- **autoplay** Hiermee wordt de video direct geladen en afgespeeld. Als u het attribuut weglaat, gebeurt dat niet (ook dit is een booleaans attribuut). Dit heeft in de meeste situaties en bij de meeste bezoekers de voorkeur.
- **preload** Dit attribuut heeft drie waarden. Met preload="auto" wordt de keus aan de browser gelaten, die al dan niet de video downloadt (dit kan per browser en per platform verschillen), maar niet direct afspeelt. Als autoplay is ingesteld, wordt preload genegeerd. Nu heeft vooraf laden alleen zin als de film bekijken moet worden. Laat u die keus aan de gebruiker, dan schakelt u met preload="none" expliciet het vooraf laden uit. Een derde optie is preload="metadata", waarbij alleen gegevens zoals afmetingen, lengte en codec worden opgehaald. Handig, want dan kan de gebruiker in de speler zien hoe lang de video duurt.
- **poster** Hiermee wordt een afbeelding getoond zolang de video wordt gedownload en niet wordt afgespeeld. De waarde van poster is de URL van een afbeelding. Als u dit attribuut niet instelt, kan het eerste frame uit de film worden getoond, maar het spelervenster kan ook zwart blijven.
- **muted** Dit booleaanse attribuut dempt het geluid van de video. De gebruiker kan met de bedieningsknoppen zelf het volume regelen.
- **loop** Booleaans attribuut voor het eindeloos herhalen van de video.

De code doet het volgende:

- er wordt naar één bestand verwezen;
- alleen metadata wordt geladen;
- er wordt niet automatisch afgespeeld (`autoplay` ontbreekt);
- er wordt een posterafbeelding getoond;
- er worden bedieningselementen (`controls`) getoond;
- het geluid is ingeschakeld (omdat het attribuut `muted` ontbreekt);
- er wordt een link aangeboden om het bestand te downloaden wanneer de browser `<video>` niet ondersteunt.

```
<video src="video/trailer_480p.mov" poster="video/html-poster.png" controls preload="metadata"
width="640" height="360">
<p>Upgrade uw browser. Download intussen <a href="video/trailer_480p.mov">de film </a>.</p>
</video>
```



Afbeelding 5.6 Het element `video` met een posterafbeelding (05_06.html).



Geen oplossing voor foute links

De downloadmededeling is geen oplossing voor niet-werkende links.

Als de browser `<video>` ondersteunt maar de film niet kan laden, ziet u een speler met een leeg scherm.

De bron: `<source>`

Gewoonlijk biedt u verschillend gecodeerde videobestanden aan om problemen zo veel mogelijk voor te zijn. De verwijzingen plaatst u in het element `<source>`. `<source>` is altijd een kind van `<video>` of `<audio>` en heeft twee attributen (naast de globale attributen):

- **src** Dit attribuut bevat de URL naar het videobestand en moet er zijn (want anders gebeurt er niks).
- **type** Aanbevolen wordt ook het attribuut type in te stellen op het juiste type video. Daarmee wordt voorkomen dat de browser videobestanden downloadt die toch niet zijn af te spelen (een niet-ondersteund bestandsformaat). Met type kan de parameter codecs worden gebruikt. Dan moet u echter wel weten met welke codec het bestand is gecodeerd. Beter geen codec dan een verkeerde codec. Een overzicht van waarden voor codecs vindt u op www.w3.org/TR/html/semantics-embedded-content.html#the-source-element. Waarden voor type kunnen zijn:
 - `video/mp4`
 - `video/ogg`
 - `video/webm`



Leeg element

`<source>` is een leeg element. Het heeft geen sluittag en u hoeft het niet af te sluiten.

Het voorgaande voorbeeld met alleen `<video>` wordt uitgebreid met `<source>` en biedt nu twee videobestanden met verschillende coderingen aan:

```
<video controls preload="metadata" width="640" height="360" poster="video/html-poster.png">
  <source src="video/trailer_480p.mov" type="video/mp4">
  <source src="video/trailer_400p.ogg" type="video/ogg"> <p>Upgrade uw browser. Intussen kunt u
    <a href="video/trailer_480p.mov">de film</a> downloaden.</p>
</video>
```



Aanhalingsstekens

Als u het attribuut type gebruikt met codecs, moeten u de aanhalingsstekens precies zo toepassen: `type='video/ogg; codecs='theora, vorbis'"'`. Het geheel tussen enkele aanhalingsstekens en de codecs tussen dubbele. De code werkt niet als u het andersom doet.

De videospeler biedt nog precies dezelfde mogelijkheden, maar nu kan de browser bepalen welk type bestand kan worden opgehaald en afgespeeld.

Ondertiteling met `<track>`

Met HTML 5 is ook de mogelijkheid geïntroduceerd om ondertitels of andere op tekst gebaseerde ondersteuning aan video toe te voegen. Verantwoordelijk daarvoor is het element `<track>`.



Testen op een webserver

Op het lokale bestandssysteem (te herkennen aan het protocol file ://) worden de bestanden met de ondertitels niet geladen. Testen kan alleen via een webserver (die wel lokaal geïnstalleerd mag zijn).

Eerst het element zelf. `<track>` is altijd een kind van `<video>`, komt altijd na de `<source>`-elementen en heeft vijf attributen:

- `kind` hiermee bepaalt u wat voor tekst geladen gaat worden. Het attribuut kan de volgende waarden hebben:
 - `subtitles` ondertitels bestaande uit een geschreven weergave of vertaling van de gesproken woord;
 - `captions` een tekstuele vervanging van al het geluid in de film, voor als het geluid uitstaat of niet hoorbaar is (omgevingsgeluid/gehoorbeperking);
 - `descriptions` beschrijving van het beeld in de film, als vervanging voor situaties waarin het beeld niet zichtbaar is – bestemd voor geluidsweergave;
 - `chapters` hoofdstuktitels, bedoeld voor navigatie en zichtbaar als interactieve lijst in de browser;
 - `metadata` bedoeld voor scripting, wordt niet weergegeven.
- `src` het adres (URL) van het titelbestand.
- `srclang` de taal waarin de titels zijn geschreven.
- `label` een naam voor het titelbestand die in het titelmenu kan verschijnen.
- `default` activeert standaard een track als de gebruiker geen andere keus heeft gemaakt. Dit is een booleaans attribuut.

Bekijk de volgende code. Er worden twee ondertitelingstracks aangegeven, een standaard geactiveerde Nederlandse en een Engelse versie.

```
<track src="subtitles.vtt" kind="subtitles" srclang="nl" label="Nederlands" default>
<track src="subtitles2.vtt" kind="subtitles" srclang="en" label="English">
```

Deze code kan als volgt in het `<video>`-voorbeeld worden opgenomen:

```
<video src="video/wildlife.mp4" poster="video/html-poster.png" controls preload="metadata"
width="640" height="360">
<track kind="subtitles" src="video/subtitles.vtt" srclang="nl" label="Nederlands" default>
<track kind="subtitles" src="video/subtitles2.vtt" srclang="en" label="English">
</video>
```



Afbeelding 5.7 Ondertitels in Chrome (05_07.html).



Geluid met ondertitels

U kunt geen ondertitels tonen met `<audio>`, domweg omdat `<audio>` geen venster genereert waarin de titels kunnen worden getoond. Wilt u gesproken woord met ondertitels aanbieden, dan biedt u de audio aan in een videobestand met ondertiteling.

Titelbestanden: WebVTT

Met alleen een verwijzing naar een bestand bent u er natuurlijk niet. Er moet ook een bestand met titels zijn. Er is bij W3C een standaard voor titelbestanden: WebVTT, wat staat voor Web Videc Text Tracks. Het is de vervanger van WebSRT (Web Subtitle Resource Tracks). WebVTT is te vinden op www.w3.org/TR/2018/CR-webvtt1-20180510/.

Een titelbestand is een gewoon tekstdocument met de extensie .vtt en een eenvoudige opbouw. Een voorbeeld:

WEBVTT

```
00:00:02.000 --> 00:00:07.000
Als we stil zijn, zien we hem misschien.
00:00:07.500 --> 00:00:11.000
Maar hij is schuw, heel schuw.
```

Een vtt-bestand moet beginnen met WEBVTT, gevolgd door een lege regel. Op de volgende regel komt de tijdcode voor de start en het eind van de titel. De

code is opgebouwd als uren:minuten:seconden.milliseconden. Uren, minuten en seconden moeten uit twee cijfers bestaan en de milliseconden (let op de punt) uit drie. Daarna komt de tekenreeks --> gevuld door de eindcode. Op de volgende regel komt de te vertonen tekst gevuld door een lege regel. Een return in de regel vertoont de titel ook over twee regels in de video. Daarna herhaalt het geheel zich zo vaak als nodig.

Hoewel de bestandsopbouw eenvoudig is, is het veel handwerk. Er is nog nauwelijks (geen?) software voor ondertiteling die kan exporteren naar WebVTT. Er is wel goede software om WebSRT-bestanden te maken en die kunnen vrij eenvoudig worden omgezet naar WebVTT. Vaak is het voldoende als u boven in het srt-bestand de tekst WEBVTT en een lege regel toevoegt, de komma die de milliseconden scheidt van de seconden vervangt door een punt en het bestand opslaat als met de extensie vtt.

Audio op de website gebruiken

Toepassing van het element voor geluid op een website, `<audio>`, heeft wat minder voeten in aarde dan het element `<video>`. De toepassing komt in grote lijnen op hetzelfde neer:

- Bied audio aan in verschillende bestandstypen, bij voorkeur mp3 en ogg.
- Bied een downloadlink aan.

Kijk voor een uitgebreide beschrijving van de attributen in de paragrafen over `<video>` en `<source>`, want die zijn hetzelfde.

Hoewel er meer audiobestandstypen kunnen worden gebruikt, bestrijkt u met mp3 en ogg wel het hele speelveld. Bied de bestanden aan in afzonderlijke `<source>`-elementen, met het juiste type. De parameter `codecs` is niet verplicht.

In het voorbeeld wordt alleen de afspeelbalk getoond.

```
<audio controls preload="none">
  <source src="audiotrack.mp3" type="audio/mpeg; codecs="mp3">
  <source src="audiotrack.ogg" type="audio/ogg; codecs="vorbis">
  <p>Download deze track als <a href="audio/audiotrack.mp3">MP3</a>
  of als <a href="audio/audiotrack.ogg">Ogg</a>.</p>
</audio>
```



Aanhalingstekens

Gebruik de aanhalingstekens bij type precies zo als in dit codevoorbeeld, dus openen met ', de codecs tussen " en sluiten met '. De code werkt niet als u het andersom doet.

Samenvatting

In dit hoofdstuk is *embedded content* besproken, inhoud die niet in de HTML-code zelf staat, maar wordt opgehaald uit een externe bron en getoond in de browser.

- De bekendste vorm van embedded content is de afbeelding. Dat kan een bitmapafbeelding zijn (GIF, JPG of PNG) of een vectorafbeelding (SVG).
- Een afbeelding wordt gemarkerd met ``. Dit element bevat een verwijzing naar de bron van het beeld (`src`) en meestal een beschrijving (`alt`).
- In een afbeelding kunnen klikbare gebieden worden gedefinieerd die elk naar een andere URL kunnen leiden.
- Het is mogelijk om afbeeldingen in verschillende resoluties en afmetingen aan te bieden, waarna de browser op basis van resolutie of schermformaat bepaalt wat de beste keus is.
- Met `<iframe>` worden complete webpagina's, een twitterfeed of andere externe bronnen ingesloten, al dan niet beveiligd.
- HTML 5 is voorzien van het element `<video>` waarmee zonder tussenkomst van andere technieken video in de webpagina wordt afgespeeld. Ook ondertiteling is mogelijk. Vanwege ondersteuningsverschillen is het nog wel raadzaam video in verschillende bestandsformaten aan te bieden.
- Eén videobron wordt gedefinieerd in het attribuut `src`. Meerdere bronnen worden opgenomen in het element `<source>`.
- Ook *native audio* (er is geen plug-in of externe software voor nodig) is beschikbaar in HTML 5. Met het element `<audio>` wordt een eenvoudige muziekspeler in de webpagina weergegeven. Met `<source>` worden de audiobronnen ingesteld.

Hiermee zijn alle elementen besproken die nodig zijn om een gestructureerde webpagina met tekst en beeld te maken. Dat betekent niet dat het HTML-deel is voltooid: u hebt nog twee onderwerpen te goed: formulieren en tabellen. Het volgende hoofdstuk gaat over formulieren.

Oefeningen



Gebruik de voorbeeldcode

Van verschillende afbeeldingen is de HTML- en CSS-code beschikbaar. De bestandsnaam staat in het bijschrift. Gebruik deze code om te oefenen: pas de code aan, voeg dingen toe, speel ermee! Kijk op handboek-html-css.nl.

- Bouw een kleine fotogalerij. Het is de bedoeling dat de bezoeker op de indexpagina miniaturen ziet, bijvoorbeeld foto's van 300 pixels breed. Zo'n miniatur linkt door naar de afbeelding op groot formaat, bijvoorbeeld 800px breed.
 - Begin met het beantwoorden van de vraag welke elementen u nodig hebt. Tip: op de pagina met miniaturen wordt `<figure>` niet gebruikt.
 - Maak in elk geval een pagina voor de miniaturen. U kunt de grote foto's laten openen op dezelfde pagina.
 - Alternatief: maak ook losse pagina's voor de grote foto's en plaats daarop een hyperlink terug naar miniaturen.
 - Voor prachtige rechtenvrije foto's kunt u bijvoorbeeld terecht op unsplash.com. Een van de vele andere mogelijkheden is pixabay.com. U kunt natuurlijk ook eigen foto's gebruiken.
 - Voor deze oefening hoeven de foto's niet per se het juiste formaat te hebben. U kunt met het attribuut `width` of `height` van het element `` het gewenste formaat instellen. Waarom moet u niet beide attributen instellen? (Dit is geen aanbevolen werkwijze, maar een praktische oplossing om te omzeilen dat u nog niet kunt werken met CSS-eigenschappen voor afmetingen.)
 - Gebruik de elementen zoals het hoort: schrijf een `alt`-tekst bij ``, maak een bijschrift in `<figcaption>`.
- Maak een HTML-pagina met daarop responsieve afbeeldingen gebaseerd op schermformaat. Dus: op een klein schermformaat wordt een kleinere afbeelding weergegeven dan op een groot schermformaat. Neem als breekpunten 600px, 1200px en 1900px. (Kies aangepaste breekpunten als uw laptop smaller is.)
 - Gebruik `` en `srcset`.
 - Test door het venster smaller en breder te maken of gebruik de device toolbar/responsive-design-modus (Ctrl+Shift+M) van de ontwikkelaarshulpmiddelen.
- Maak een HTML-pagina met een `<header>` en daarin een afbeelding. Zorg ervoor dat op verschillende schermenbreedten een andere versie van de afbeelding worden getoond met een uitsnede die past bij de schermbreedte.
 - Gebruik `<picture>`, `<source>` en ``.

- Kies of maak liggende afbeeldingen met een hoogte van 300px en verschillende breedten: 360px, 700px, 1024px en 1920px.
- Tip: voeg de volgende code toe als attribuut van `` in het element `<picture>`. Hiermee wordt de afbeelding altijd net zo breed als het venster. (Dit kan niet met het attribuut `width` van ``, want dat accepteert alleen een getal zonder eenheid dat staat voor pixels.)

```

<ul>
  <li>
    <label for="text"><input type="text"></label>
    <input id="text" type="text" name="text" autofocus>
  </li>
  <li>
    <label for="email"><input type="email"></label>
    <input id="email" type="email" name="email">
  </li>
  <li>
    <label for="tel"><input type="tel"></label>
    <input id="tel" type="tel" name="tel" maxlength="10" placeholder="alleen cijfers">
  </li>
  <li>
    <label for="url"><input type="url"></label>
    <input id="url" type="url" name="url">
  </li>
  <li>
    <label for="password"><input type="password"></label>
    <input id="password" type="password" name="password">
  </li>
```

```
<li>
    <label for="number">input type="number"
    </label>
    <input id="number" type="number" name="number" value="0" step="5">
</li>
<li>
    <label for="range">input type="range"
        <small>-100 tot 100, stappgroote 10</small>
    </label>
    <input id="range" type="range" name="range" value="0" min="-100" max="100" step="10">
</li>
<li>
    <label for="color">input type="color"</label>
    <input id="color" type="color" name="color" value="#66bb77">
</li>
<li>
    <label for="date">input type="date"</label>
    <input id="date" type="date" name="date">
</li>
<li>
    <label for="time">input type="time"</label>
    <input id="time" type="time" name="time">
</li>
<li>
    <label for="datetime-local">input type="datetime-local"</label>
    <input id="datetime-local" type="datetime-local" name="datetime-local">
</li>
<li>
    <label for="week">input type="week"</label>
    <input id="week" type="week" name="week">
</li>
<li>
    <label for="month">input type="month"</label>
    <input id="month" type="month" name="month">
</li>
<li>
    <label for="checkbox">input type="checkbox"</label>
    <fieldset id="checkbox">
        <legend>Vergroot het comfort van uw e-bike met:</legend>
        <label for="opt1">
            <input id="opt1" type="checkbox" name="comfort" value="airco">airconditioning</label>
        <label for="opt2">
            <input id="opt2" type="checkbox" name="comfort" value="stuurverwarming">stuurverwarming</label>
    
```

```
<label for="opt3">
    <input id="opt3" type="checkbox" name="comfort"
        value="zadelverwarming">zadelverwarming</label>
</fieldset>
</li>
</li>
<label for="checkboxx"><input type="radio"></label>
<fieldset>
    <legend>Aantal medewerkers</legend>
    <label for="opt4"><input id="opt4" type="radio" name="mw" value="1">1 (alleen ik)</label>
    <label for="opt5"><input id="opt5" type="radio" name="mw" value="2-10">2 - 10</label>
    <label for="opt6"><input id="opt6" type="radio" name="mw" value="11-50">11 - 50</label>
</fieldset>
</li>
<li><label for="gezond">De gezonde keus</label>
    <select id="gezond" name="gezond">
        <optgroup label="groenten">
            <option value="spinazie">spinazie</option>
            <option value="witte kool">witte kool</option>
            <option value="tuinbonen">tuinbonen</option>
        </optgroup>
        <optgroup label="fruit">
            <option value="appel">appel</option>
            <option value="peer">peer</option>
            <option value="mandarijn">mandarijn</option>
        </optgroup>
    </select>
</li>
<li>
    <label for="genre-lijst">Welke muziekstijl wil je horen?</label>
    <input list="genre" id="genre-lijst" name="stijl">
    <datalist id="genre">
        <label for="lijst-fail">Kies een muziekstijl:</label> <!-- Terugvalmechanisme voor
            iOS/Safari -->
        <select id="lijst-fail" name="stijl"> <!-- Terugvalmechanisme voor iOS/Safari -->
            <option value="pop">
            <option value="rock">
            <option value="indie">
            <option value="dance">
        </select>
    </datalist>
</li>
<li>
    <label for="mening">Opmerkingen bij uw bestelling:</label>
```

```

<textarea id="mening" name="mening" rows="5"></textarea>
</li>
</ul>
<input type="reset" value="Herstellen">
<input type="submit" value="Versturen">
</form>

```

<input type="text"/>	
<input type="email"/>	
<input type="tel"/>	010 12345678
<input type="url"/>	
<input type="password"/>	
<input type="number"/>	0
<input type="range"/>	100 tot 100, stepgrootte 10
<input type="color"/>	#000000
<input type="date"/>	dd-mm-yyyy
<input type="time"/>	----
<input type="datetime-local"/>	dd-mm-yyyy -- --
<input type="week"/>	week -- --
<input type="month"/>	-----
<input type="checkbox"/>	Vergroot het comfort van uw e-bike met:
	<input type="checkbox"/> airconditioning <input type="checkbox"/> stuurverwarming <input type="checkbox"/> zadelverwarming
<input type="radio"/>	Aantal medewerkers
	<input type="radio"/> 1 (alleen ik) <input type="radio"/> 2 - 10 <input type="radio"/> 11 - 50
De gezonde keus	<input type="button" value="Spinazie"/>
Welke muziekstijl wil je horen?	
Opmerkingen bij uw bestelling:	
	<input type="button" value="Herstellen"/>
	<input type="button" value="Versturen"/>

Afbeelding 6.3 Dit HTML-formulier bevat een groot deel van de beschikbare elementen (06_03.html).

De basis van een formulier: <form>

Het element <form> is de basis van een formulier. Alle inhoud van dat element wordt gezien als formulierinhoud. <form> heeft attributen die de webserver vertellen hoe het formulier moet worden verwerkt: action, method en enctype. Verwerking kunt u ruim zien, want die kan bestaan uit het sturen van een reactie per mail naar de inzender, het opslaan van de gegevens in een database tot het volledig geautomatiseerd afhandelen van een order. Dat gebeurt uiteraard allemaal op de server en valt buiten het bereik van HTML. Daarnaast zijn er attributen voor automatisch aanvullen, validatie en de naam van het formulier.

Buiten <form> mag ook

Invoerelementen en knoppen mogen ook buiten het element <form> staan. Er zijn attributen waarmee bijvoorbeeld een tekstveld of een verzendknop 'ergens' in de HTML-pagina kan staan en toch kan worden gekoppeld aan een formulier. Hierbij spelen het attribuut form op het invoerelement en de id van het <form> een sleutelrol. Daarnaast zijn er attributen waarmee de verwerking van dergelijke elementen kan worden ingesteld. Omdat de werking van deze attributen duidelijker is als u weet hoe in het algemeen een formulier wordt verstuurd, bekijken we eerst attributen van het element <form>.

De verwerking: action

In het attribuut action wordt aangegeven naar welke URL het ingevulde formulier moet worden verstuurd. Voorbeelden zijn een CGI-script of een PHP-pagina op de webserver: <form action="/verwerk.cgi"> of <form action="https://www.domein.nl/verwerk.php">.

Verzendwijze: method

De ingevulde gegevens van een formulier kunnen op twee manieren naar de webserver worden gestuurd. Om het verschil te kunnen begrijpen, moet u iets weten over hoe een HTTP-aanvraag werkt. Voor het raadplegen van een bron op internet stuurt de browser een aanvraag naar het webadres (de URL). Die aanvraag wordt een HTTP-request genoemd. De aanvraag bevat twee dingen: een header met informatie over de browser en een body met informatie die de server nodig heeft om de aanvraag te kunnen afhandelen.

De methode GET

Het element <form> heeft het attribuut method dat de verzendwijze bepaalt. De opties zijn GET en POST (get en post mag ook). Bij de methode GET worden

de formuliergegevens toegevoegd aan de URL waarmee het formulier wordt verstuurd. Bij een formulier met velden voor een naam en een telefoonnummer zou de URL er zo uit kunnen zien:

```
https://www.domein.nl/verwerk.cgi?naam=Doolaard&telnr=0698765432
```

Dit brengt gelijk de twee beperkingen van GET aan het licht: alle formuliergegevens worden open en bloot verstuurd en bij een formulier van enige omvang kan de limiet op de lengte van een URL worden overschreden. Die varieert, maar ligt rond 2000 tekens. GET is ook niet bedoeld voor het verzenden van grote hoeveelheden data, maar voor het ophalen van bijvoorbeeld resultaten.

De methode POST

In de meeste gevallen heeft de methode POST de voorkeur. Het verschil met GET is dat de formuliergegevens niet in de URL, maar in de body van de HTTP-aanvraag worden geplaatst. Daarmee zijn ze én niet zichtbaar én er past veel meer informatie in. Vandaar ook dat dit voor het meesturen van bestanden de enige methode is.

Codering: enctype

Dit attribuut wordt alleen ingesteld wanneer ook bestanden worden geüpload. Er zijn drie mogelijke waarden:

- `application/x-www-form-urlencoded`, de standaardwaarde;
- `multipart/form-data`, de waarde als ook bestanden worden geüpload;
- `text/plain`.

De instelling bepaalt welk deel van de inhoud wordt gecodeerd (respectievelijk alles, niets of alleen spaties) en dat is belangrijke informatie voor de ontvangende server, omdat die anders de gegevens niet begrijpt. In veel situaties volgt de standaardwaarde en hoeft u enctype niet in te stellen.

Automatisch aanvullen: autocomplete

Het attribuut `autocomplete` stelt u alleen in als u *niet* wilt dat velden automatisch worden aangevuld. Standaard onthouden browsers wat de gebruiker bij bepaalde velden invult en bieden die gegevens een volgende keer weer aan. Met `autocomplete="off"` schakelt u dit uit. Dit mechanisme kunt u niet volledig controleren, omdat het wachtwoordbeheer van de browser met toestemming van de gebruiker evengoed inloggegevens kan opslaan als autocomplete uit staat. (Dit geldt ook voor afzonderlijke invoervelden voor gebruikersnaam en wachtwoord.) Voor andere gevoelige informatie (creditcardcode, burgerservicenummer) of gegevers die een volgende keer anders zullen zijn (captcha,

tijdcode) kunt u het wel gebruiken. Auto-aanvullen hoeft niet per se op het formulier te worden uitgeschakeld; het kan ook op afzonderlijke invoervelden worden uitgeschakeld.

Als autocomplete="off" is ingesteld op het formulier, moet een gebruiker een volgende keer (ook bij een vergelijkbaar formulier) alle gegevens opnieuw invullen. Die worden ook niet in de cache opgeslagen, dus bij terugklikken naar het begin van het formulier zijn de velden leeg.

Zie ook de uitleg bij het attribuut autocomplete van het element <input>.

Niet valideren: novalidate

De ingebouwde controle van formuliergegevens is zonder meer handig, maar soms kan die in de weg zitten, bijvoorbeeld als u zelf met JavaScript wilt valideren. Wilt u de controle uitschakelen, dan kan dat met het attribuut novalidate op het element <form>. Het is een booleaans attribuut (het heeft dus geen waarden).

Het gelijkwaardige attribuut formnovalidate kan ook op andere formulierelementen worden gebruikt en dat opent mogelijkheden, terwijl u toch het risico op een verkeerd of onvolledig ingevuld formulier beperkt. Een voorbeeld is een uitgebreid formulier (bijvoorbeeld een enquête) waarvan u kunt vermoeden dat de bezoeker die niet in één keer zal invullen. Dan is een optie om het formulier tussentijds te kunnen opslaan wel zo vriendelijk. Maar als het formulier verplichte velden bevat of velden die een bepaalde invoer vereisen en die velden zijn nog leeg, dan staat de ingebouwde validatie niet toe dat het formulier wordt verstuurd. Door nu een verzendknop toe te voegen voor tussentijds opslaan en die als attribuut formnovalidate te geven, kan het tussentijdse resultaat toch worden verzonden.

Ter illustratie de code van een gewone verzendknop en een knop voor tussentijds opslaan (<input> wordt verderop behandeld):

```
<input type="submit" name="verzenden" value="Verzenden, het formulier is ingevuld">
<input type="submit" formnovalidate name="verzenden" value="Opslaan en later doorgaan">
```

Bedenk dat dit slechts een deel van het verhaal is (het HTML-deel), want u zult ook voorzieningen moeten treffen voor het opnieuw ophalen van het formulier met de al ingevulde gegevens.

Direct invoeren: autofocus

Geen attribuut van `<form>` maar van de afzonderlijke invcerelementen: `autofocus`. Dit attribuut zorgt ervoor dat bij het openen van de pagina met het formulier de cursor direct in een invoerveld staat. Het is een booleaanse attribuut. Het resultaat van `autofocus` op een invoerveld is dat als u na het openen van de pagina blindelings begint te typen, de informatie ook direct terechtkomt in het verwachte veld. Dat laatste is belangrijk. U moet wel zeker weten dat uw bezoekers het logisch vinden dat precies dat veld de focus heeft. Als dat niet ondubbelzinnig is, schiet het hulpmiddel voor een deel van de bezoekers zijn doel voorbij en kunt u het beter weglaten.



Visuele hint met CSS

De meeste browsers geven een visuele hint wanneer een veld de focus heeft, ook bij `autofocus`. De opmaak daarvan kan met CSS worden aangepast aan de stijl van de pagina.

De naam: name

Het attribuut `name` bevat de naam van het formulier. Deze naam is voornamelijk van belang als u het formulier met scripting wilt benaderen. De namen moeten uniek zijn als de pagina meerdere formulieren bevat.



Koppelen aan id

Invoerelementen die buiten het formulier staan, worden gekoppeld aan de `id` van het formulier, niet aan de `name`.

Het doel: target

Het resultaat van het formulier kan worden weergegeven in dezelfde of een andere pagina. De standaardactie is dezelfde pagina als die het formulier bevat, maar met het attribuut `target` kunt u een ander doel instellen, bijvoorbeeld `_blank` voor een nieuwe pagina.

Buitenspelelementen: form

Hiervoor (in de paragraaf *Buiten `<form>` mag ook*) is uitgelegd dat invoerelementen niet per se een afstammeling van het element `<form>` hoeven te zijn. Ze mogen ook buiten het element staan, bijvoorbeeld in een tabel ergens anders op de pagina.

Om de informatie van dat invoerelement of eventueel een speciale verzendknop te koppelen aan het juiste formulier, zijn er twee dingen nodig:

- het formulier moet een id hebben;
- het invoerelement of de knop heeft het attribuut form met als waarde de formulier-id.

De tag <form> ziet er bijvoorbeeld zo uit:

```
<form action="verwerk.php" method="post" id="verzamel">
```

Ergens anders op de pagina – buiten het element <form> – staat bijvoorbeeld een tekstveld:

```
<input form="verzamel" name="mening">
```

In dit tekstveld kan de bezoeker zijn mening geven. Het veld is met form="verzamel" gekoppeld aan het formulier met id="verzamel" en als dat formulier wordt verstuurd, gaat de inhoud van dit tekstveld vanzelf mee.

Op een vergelijkbare manier kan een alternatieve verzendknop worden gemaakt, die een ander verwerkingsscript aanroept. In de code hiervoor heeft het formulier de action="verwerk.php". Met de volgende knop wordt hetzelfde formulier verstuurd, maar met een ander script. Daarvoor wordt het attribuut formaction gebruikt:

```
<input form="verzamel" type="submit" formaction="verwerk_anders.php">
```



Alle attributen

Behalve formaction zijn er ook de attributen formenctype, formmethod en formtarget, waarmee de waarden van enctype, method en target die zijn ingesteld in het element <form> worden overschreven. Met formnovalidate kan de validatie van de invoer worden omzeild.

Formulierelementen die buiten het element <form> kunnen worden gebruikt:

<button>	<keygen>	<select>
<fieldset>	<label>	<textarea>
<input>	<output>	

Labels bij invoervelden

Met alleen een element `<form>` en wat verzendinstructies begint u weinig. Er moet wel wat in te vullen zijn. Wat dat betreft kunt u uw hart ophalen, want HTML 5 biedt zo'n twintig invoertypen variërend van een tekstveld, via een datumkiezer en bestandsupload, tot een verzendknop en daarbij nog zo'n tien elementen waarmee u leuke dingen kunt doen. Het hoofdstuk is nog niet uit, zullen we maar zeggen.

Voordat al die invoertypen en elementen aan bod komen, eerst iets meer over de algemene opbouw van een formulier. Van elk invoerveld moet duidelijk zijn welke invoer wordt verwacht en dat bewerkstelligt u door een bijschrift of label toe te voegen. Dan rijst de vraag hoe u een label met zijn invoerveld (en eventueel andere bijbehorende onderdelen) bundelt tot een logische eenheid. Als u het antwoord op internet zoekt, vindt u allerlei visies. De een vindt dat een formulier een soort lijst is, waarbij de bij elkaar horende formulierelementen in een lijst-item `` worden geplaatst. Een andere visie is dat die elementen in een `<div>` kunnen worden gebundeld. Een derde visie is dat elk onderdeel van een formulier een nieuwe alinea is en daarom kan worden opgenomen in het element `<p>`. Het aardige is dat er voor elk van deze benaderingen wel een toepassing is: een reeks keuzerondjes of selectievakjes als lijst, labels en invoervelden in een alinea en worden het meer elementen, dan kunnen ze in een `<div>` worden gebundeld. Hoewel, alleen bundelen voor opmaak is dankzij nieuwe lay-outtechnieken zoals flexbox en gridlay-out ook niet meer per se nodig. Een `<div>` voegt geen betekenis toe, dus als u zonder kunt is dat prima. Bedenk dat er geen wet is die voorschrijft dat en hoe elementen in een formulier moeten worden gebundeld. En inzichten kunnen veranderen. Bij HTML 5 adviseerde het W3C de `<p>` als wrapper, terwijl sinds HTML5.1 de `<div>` wordt gebruikt. Het belangrijkste is dat u een oplossing kiest die duidelijk en toegankelijk is.



Formulier als tabel

Een formulier kan ook bestaan uit een tabel. Denk aan een overzicht van te bestellen producten, met kolommen voor de productnaam, de prijs en het te bestellen aantal.

In `<label>` kunnen ook aanwijzingen voor het invullen van het veld worden opgenomen. Omdat een aanwijzing een toevoeging is aan het label, is het element `<small>` een geschikte oplossing:

```
<label>Postcode: <small>1234 AB</small>
  <input id="pc" name="pc">
</label>
```

In de voorgaande code omvat het label zowel de bijschrifttekst als het invoerveld. Het label mag echter ook apart staan. Met het attribuut `for` blijft de koppeling met het bijbehorende invoerelement in stand, zoals in het volgende fragment.

```
<div>
  <label for="anaam">Uw achternaam:</label>
  <input id="anaam" name="anaam">
</div>
```

Het loskoppelen van `<label>` en `<input>` heeft twee voordeelen: de code wordt overzichtelijker en de elementen zijn veel eenvoudiger te beraderen en op te maken met CSS. Met name uitlijning gaat een stuk gemakkelijker (al speelt dat eigenlijk dankzij flexbox en gridlay-out geen rol meer).

In het tweede voorbeeld krijgt het invoerelement ook de focus als het label wordt geselecteerd. Dat komt door de koppeling tussen beide met het attribuut `for`, maar ook zonder dat bijeffect wordt aanbevolen om het attribuut `for` te gebruiken, omdat daardoor de relatie duidelijk is.

De tabvolgorde

Wat op een formulier goed fout kan gaan, is de tabvolgorde. Dat wil zeggen de volgorde waarin de velden de focus krijgen als de gebruiker op de Tab-toets drukt. Zolang u het formulier opbouwt in de volgorde waarin de velden worden weergegeven, is er niets aan de hand en gaat het vanzelf goed. Gebruikt u echter CSS (Flexbox of Grid Layout) of JavaScript om de HTML in een ander volgorde te laten weergeven, dan zal navigatie met de Tab-toets de volgorde van de HTML-code volgen, niet die van de manier waarop de invoervelden worden gepresenteerd. U zult dan met het globale attribuut `tabindex` de logica terug moeten brengen, want niets is zo verwarrend als een formulier waarin de gebruiker met de Tab-toets van het naar het wordt gestuurd. Dat is een ongewenste situatie en het heeft de voorkeur ervoor te zorgen dat de codevolgorde gelijk is aan de weergegeven volgorde.

Het element `<input>`

Het elementaire invoerdeeltje is `<input>`. Zonder attributen is het een veld waarin (korte) tekst zonder regeleinden kan worden ingevoerd (druk op Enter en er zal niets gebeuren). Met het attribuut `type` kan `<input>` veranderen in een schuifregelaar, kleurkiezer of keuzerondje en nog veel meer, waarover we het hierna zullen hebben.

De standaardwaarde is type="text" en dat is ook de terugvalwaarde. Als type een voor de browser onbekend waarde heeft (niet ondersteund of doordat u een typefout hebt gemaakt), wordt altijd een tekstveld weergegeven. Vanwege die zekerheid kunt u veilig elk nieuw invoertype gebruiken, want in het slechste geval krijgt de bezoeker altijd nog een tekstveld voorgesloten dat hij kan invullen.



Invoer beperken met pattern

U kunt zelfs nog een terugvalmechanisme toevoegen voor browsers die een bepaald invoertype niet ondersteunen. Het verderop besproken attribuut pattern biedt de mogelijkheid om met een reguliere expressie nauwkeurig te bepalen welke invoer geldig is voor een veld.

Attributen van <input>

De werking van de diverse typen van <input> wordt aangepast met attributen. Naast de globale attributen heeft <input> ruim dertig eigen kenmerken. Die worden niet allemaal besproken; de uitleg wordt beperkt tot de meest gebruikte attributen.

Opdracht

Experimenteer tijdens het lezen met de attributen om te zien wat het resultaat is van de code en wat de in de browser ingebouwde validatie met foute invoer doet. Maak daarvoor een minimaal formulier met een verzendknop en voeg de besproken invoertypen toe. Hier is een <div> gebruikt als wrapper, omdat dan de setjes van een <label> en een <input> onder elkaar komen te staan.

```
<form action="">
  <div>
    <label for="veld">Labeltekst:</label>
    <input id="veld" name="naam" type="type" attributen>
  </div>
  <input type="submit">
</form>
```

Onmisbaar: name

Een invoerveld zonder naam is ondenkbaar. Elk type invoerveld moet een naam hebben, anders kan de waarde nergens aan worden gekoppeld. Bij het verzenden van een formulier worden die naam en waarde namelijk als paar toegevoegd aan de HTTP-aanvraag: vnaam=Peter&anaam=Doolaard.

Het attribuut name mag elke waarde hebben (maar _charset_ en index zijn namen met een speciale functie die hier buiten beschouwing blijven).

De verstuurde waarde: value

Het attribuut `value` bevat de waarde van een invoerveld, die wordt verstuurd als het formulier wordt ingediend. Bij tekstinvoer wordt de waarde automatisch ingesteld door de invoer van de gebruiker en bij invoertypen zoals datum (kalender) of kleur (kleurkiezer) door de keuze van de gebruiker in die interface. In die gevallen hoeft u `value` niet te gebruiken.

Met `value` kunt u vooraf de waarde van een invoerveld instellen. Wat voor waarde dat is, hangt af van het type invoerveld. Het kan tekst zijn (eerder ingevulde klantgegevens), een getal (minimale afname van een product), een datum, tijdstip of een kleurwaarde.

Een knop (bijvoorbeeld `type="submit"`) bevat `value` het opschrift van de knop.

```
<input type="submit" value="Bestellen">
```

Het attribuut is bedoeld als hulpmiddel voor de gebruiker en bevat een waarde waarvan het waarschijnlijk is dat die zou worden ingevuld (zoals naam, adres en woonplaats van een terugkerende klant). Het is nadrukkelijk niet bedoeld om aanwijzingen voor het invullen te geven, daar is `placeholder` voor (zie verderop). Als de gebruiker namelijk het veld over het hoofd ziet en de waarde niet aanpast, krijgt u het formulier terug met die aanwijzingtekst. (Ook als het veld verplicht is, want het is immers niet leeg.)

Gelijk in het veld typen: autofocus

Eén invoerveld kan het attribuut `autofocus` krijgen. Dat heeft als resultaat dat de cursor gelijk in een tekstveld staat of – bij andere typen – het element anderszins klaar is voor invoer. Het is een booleaans attribuut.

```
<label for="vnaam">Voornaam:</label>
<input id="vnaam" name="vnaam" type="text" autofocus>
```

Grenzen aan de invoer: minlength en maxlength

Deze attributen bepalen de minimale en maximale lengte van de invoer. Een compleet bankrekeningnummer is 18 cijfers lang, zonder spaties. In code wordt dat:

```
<label>IBAN:</label>
<input minlength="18" maxlength="18">
```

Een te korte invoer geeft een foutmelding als wordt geprobeerd het formulier te verzenden. Een te lange invoer is domweg niet mogelijk; de invoer houdt gewoon op. Als de invoer niet verplicht is (`required`), protesteert de browser

niet tegen een leeg invoervak. Foutbestendig is dit allemaal niet, want letter en cijfers kunnen willekeurig worden ingevoerd. Een mogelijke oplossing biedt het attribuut `pattern`.

IBAN:

i Breid deze tekst uit tot 16 tekens of meer (je gebruikt momenteel 16 tekens).

Afbeelding 6.4 Te weinig invoer, maar of het verder klopt weet niemand.

Controle op invoer: pattern

Het attribuut `pattern` biedt de mogelijkheid om een reguliere expressie te gebruiken waarmee de invoer van het veld wordt vergeleken. Zo kunt u precies aangeven hoe de invoer moet zijn opgebouwd, bijvoorbeeld of een e-mailadres een @-teken bevat en of een domein (eindigt op een punt gevolgd door twee of drie tekens: .nl, .com). Komt de invoer daarmee niet overeen, dan komt het veld niet door de controle. Het is raadzaam om de gebruiker goede instructies te geven voor de gewenste invoer, zeker als het specifieke patronen betreft.

De HTML-specificatie adviseert een beschrijving van het patroon op te nemen in het attribuut `title` van het element `<input>`, eventueel aangevuld met een zichtbare beschrijving. Dat laatste is een goed idee, omdat smartphones geen tooltips kunnen weergeven.



Reguliere expressie

Een reguliere expressie is een manier om patronen te beschrijven. In HTML wordt daarvoor de JavaScript-standaard gebruikt. Reguliere expressies kunnen er ingewikkeld uitzien (en dat ook zijn!), maar een-eenvoudige vergelijkingen zijn goed te doen. Raadpleeg bijvoorbeeld developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions of www.regex101.com voor uitleg en voorbeelden. De tweede site biedt ook een handig testhulpmiddel.

Een correcte postcode bestaat uit vier cijfers, een spatie en twee hoofdletters. Een reguliere expressie daarvoor is: `[0-9]{4}\s[A-Z]{2}`. Het invoerveld met hulp voor de gebruiker ziet er dan zo uit:

```
<label for="pc">Postcode: <small>1234 AB</small></label>
<input id="pc" pattern="[0-9]{4}\s[A-Z]{2}" title="Een postcode bestaat uit 4 cijfers, 1 spatie en 2 hoofdletters">
```



Wees duidelijk over verwachte invoer

Laat bezoekers niet raden naar de vorm waarin u invoer verwacht, zeker als u afwijkt van de norm (zoals een postcode zonder spatie). Geef altijd een toelichting, zodat velden in een keer goed kunnen worden ingevuld.

Postcode: 1234 AB

Zorg dat de indeling voldoet aan de gevraagde indeling.
Een postcode bestaat uit 4 cijfers, een spatie en 2 hoofdletters.

Afbeelding 6.5 De invoer komt niet overeen met het patroon. Het eerste deel van de melding komt van de browser.

De tabel toont wat basiselementen om zelf reguliere expressies te maken.

[abc]	een teken uit de lijst
[^abc]	geen teken uit de lijst
[A-Z]	elke hoofdletter van A tot Z
[a-z]	elke kleine letter van a tot z
(a b)	of a of b
[0-9]	elk getal van 0 tot 9
{x}	het voorgaande moet x keer voorkomen
\s	een spatie
\S	elk ander teken dan een spatie
n+	minstens 1 teken n
n*	0 of meer keer het teken n
n?	0 of 1 keer teken n
.	1 teken (een punt is \.)
\d	een cijfer
\D	elk ander teken dan een cijfer
^	begin van de tekenreeks
\$	eind van de tekenreeks

Met het volgende patroon kan het IBAN-patroon worden gecontroleerd.

```
<label for="iban2">IBAN: <small>NL00AAAA1234567890</small></label>
<input id="iban2" name="iban" pattern="^[\A-Z]{2}[0-9]{2}\s[0-9]{4}\s[0-9]{4}\s[0-9]{2}" title="Een IBAN bestaat uit 2 letters, 2 cijfers, 4 letters en 10 cijfers">
```

Het is overigens vriendelijker voor de gebruiker om het IBAN te splitsen en die samen te voegen tot één waarde. Zo werkt bijvoorbeeld de NL IBAN hulp.

Rekening	<input type="text"/>	<input type="checkbox"/> NL IBAN hulp
Rekening	<input type="text"/>	<input checked="" type="checkbox"/> NL IBAN hulp

Afbeelding 6.6 *Aparte invoervelden voor elk onderdeel van het IBAN. Een dergelijke oplossing is geschikt voor elke andere lange invoer.*

De lengte van de invoer: size

De lengte van het invoerveld op het formulier kan worden ingesteld met het attribuut `size`. Bij de typen `text` en `password` betreft `size` het aantal tekens, anders is het een pixelwaard. Het attribuut werkt alleen bij de typen `text`, `password`, `search`, `tel`, `url`, `email`. Het attribuut beperkt niet het aantal tekens dat de gebruiker kan invoeren. Het veld kan alleen korter worden getoond. Opmaak met CSS overschrijft dit attribuut. Het ligt sowieso meer voor de hand om de lengte van invoervelden in te stellen met CSS.

Gebruiksbeperking met `readonly` en `disabled`

Een veld dat wel moet worden getoond maar niet door de gebruiker mag worden veranderd, kan `readonly` (alleen-lezen) worden gemaakt. Dit is een booleaans attribuut. Het kan bijvoorbeeld nuttig zijn in een bestelformulier waarin het artikel al is gekozen en alleen het aantal nog moet worden bepaald (number en min worden verderop uitgelegd).

```
<label for="prod">Gekozen: </label>
<input id="prod" name="prod" readonly value="Flintenladdertje">
<label for="aantal">Aantal: </label>
<input id="aantal" name="aantal" type="number" min="1">
```

Een alleen-lezenveld blijft in de tabvolgorde, kan de focus krijgen, de inhoud kan worden gekopieerd en het wordt meegestuurd als het formulier wordt ingediend; het kan alleen niet worden bewerkt. `readonly` kan alleen op tekstvoertypen worden ingesteld.

Een veld kan ook `disabled` zijn, dan is het uitgeschakeld. Het krijgt niet de focus, is niet te kopiëren en wordt niet meegestuurd met het formulier. Elk formulierelement kan worden uitgeschakeld met `disabled`. Een veelgebruikte toepassing is het uitschakelen van de verzendknop zolang niet alle vereiste onderdelen van het formulier zijn ingevuld. Voor het in- en uitschakelen van een element is JavaScript nodig.

```
<label for="prod2">Product: </label>
<input id="prod2" name="prod2">
<label for="aantal2">Aantal: </label>
<input id="aantal2" name="aantal2" disabled placeholder="Kies eerst een product">
```

Er is een duidelijk verschil tussen de twee. `disabled` is bedoeld om een veld tijdelijk uit te schakelen, bijvoorbeeld totdat de bezoeker een bepaalde keus heeft gemaakt. Van een `readonly`-veld is het nooit de bedoeling dat het wordt ingevuld of aangepast, zoals in het voorbeeld van het bestelformulier.

Verplichte invoer: required

Een verplicht veld krijgt het attribuut `required`. Handig voor inlogpagina's, waarop een gebruikersnaam en een wachtwoord nogal onmisbaar zijn, maar zeker ook voor noodzakelijke persoonlijke gegevens bij bestellingen. Het is een booleaans attribuut. Als een verplicht veld niet wordt ingevuld, verhindert de ingebouwde validatie het versturen van het formulier en zal de browser aangeven dat het veld moet worden ingevuld. Het is wel zo vriendelijk om de bezoeker vooraf duidelijk te maken dat een veld verplicht is, bijvoorbeeld door de tekst 'verplicht' of een sterretje * in het label of een kleuraccent op het veld.

```
<p><label>Straatnaam: <input required><small> (noodzakelijk)</small></label></p>
```

Straatnaam: *

Vul dit veld in.

Afbeelding 6.7 Zonder straatnaam (of in elk geval invoer in dit veld) wordt het formulier niet verzonden.

Meer waarden met multiple

Het attribuut `multiple` maakt het mogelijk meer waarden in te voeren, afhankelijk van het type invoerveld e-mailadressen of bestanden (zie ook de desbetreffende paragrafen over `<input>`-typen). `multiple` is een booleaans attribuut. (`multiple` kan ook worden gebruikt bij het element `<select>`. Zie voor uitleg de paragraaf *Kiezen uit een lijst: <select>, <datalist>, <option>*.)

Helptekst met placeholder

Een aanwijzing voor het gebruik van een veld maakt u met `placeholder`. De waarde van dit attribuut wordt in het veld getoond. Het is niet geschikt voor een uitleg in romanform, maar een voorbeeld van de gewenste invoer past natuurlijk uitstekend. De plaatshoudertekst verdwijnt zodra het veld wordt ingevuld.

```
<label for="pc3">Postcode:</label>
<input id="pc3" pattern="[0-9]{4}\s[A-Z]{2}" title="Een postcode bestaat uit 4 cijfers, 1 spatie en 2 hoofdletters" placeholder="1234 AB">
```

Vanuit het oogpunt van bruikbaarheid en toegankelijkheid is placeholder een no-gozone. De kans is groot de gebruiker niet doorheeft dat het geen invoer maar een aanwijzing is en daarom het veld niet invult. Juist omdat de plaatshoudertekst verdwijnt zodra het veld wordt ingevuld, is ook gelijk de hulp uit beeld; dat kan een probleem zijn. Het grootste bezwaar is dat screenreaders de placeholder (meestal) niet voorlezen. Een aanwijzing kan daarom beter (als -tekst) in het label worden opgenomen.

Grenzen bepalen met min en max

Het attribuut `min` geeft aan wat de kleinste waarde in een bereik is. `max` geeft de grootste waarde aan. Deze attributen worden gebruikt om de keuzemogelijkheden in een bereik af te bakenen en zijn daardoor alleen te gebruiken bij de invoertypen `number`, `range` en de typen voor datum en tijd.

De volgende code beperkt het aantal te bestellen kaartjes per klant tot 3, met een minimum van 1, waarbij 1 de standaardwaarde is (`value`). Meer of minder is niet mogelijk.

```
<label for="kaartjes">Aantal kaartjes: <small>1-3</small></label>
<input id="kaartjes" name="kaartjes" type="number" value="1" min="1" max="3">
```

Stapgrootte aanpassen met step

Ook het gebruik van het attribuut `step` is beperkt tot de invoertypen `number`, `range` en de typen voor datum en tijd. Standaard worden de waarden van `number` en `range` met 1 aangepast; met `step` kunt u zelf de grootte van de stappen instellen. Voor `date` is de stapgrootte 1 dag, voor `time` 60 seconden. Met `step="any"` is elke stapwaarde toegestaan.

list

Het attribuut `list` wordt gebruikt in combinatie met het element `<datalist>`. Zie voor uitleg de paragraaf *Kiezen uit een lijst: <select>, <datalist>, <option>*.

Automatisch aanvullen: autocomplete

Eerder kwam het attribuut `autocomplete` aan bod voor het formulier. Een attribuut met dezelfde naam is beschikbaar voor het element `<input>`. Het kan gebruikers van het formulier helpen bij het invullen door de mogelijkheid om op basis van gestandaardiseerde waarden formulievelden automatisch aan te vullen. Voor- en achternaamvelden worden dan bijvoorbeeld:

```
<label for="vnaam">Voornaam:</label>
<input id="vnaam" name="vnaam" type="text" autocomplete="given-name">
<label for="anaam">Achternaam:</label>
<input id="anaam" name="anaam" type="text" autocomplete="family-name">
```

De lijst met mogelijke waarden is lang en – met toelichting – terug te vinden in de specificatie op www.w3.org/TR/html/sec-forms.html#autofilling-form-controls-the-autocomplete-attribute.

Automatische hoofdletters: `autocapitalize`

Voornamen en de achternaam krijgen automatisch een hoofdletter met `auto-capitalise`. Dat wordt bereikt met:

```
<input autocapitalize="words">
```

Andere waarden zijn:

- `characters` maakt van alle tekens hoofdletters
- `sentences` hoofdletters aan het begin van zinnen.

Soorten invoer: het attribuut type

Behalve het standaardtype `type="text"` biedt `type` nog ruim 15 mogelijkheden voor diverse soorten invoer. Waarom zou u ze gebruiken?

- De betekenis van de invoer is gelijk duidelijk.
- Het uiterlijk van het invoerveld wordt zo aangepast dat het voor de gebruiker duidelijker is welke invoer wordt verwacht, ook zonder CSS.
- De ingebouwde controle (validatie) helpt foute invoer te voorkomen.
- Daarvoor geschikte apparaten (mobieltje, tablet) kunnen een bijpassend toetsenbord tonen.

Genoeg zinnige reden om u te verdiepen in de invoertypen; bovendien is het gewoon leuk om ermee te werken.

Telefoonnummers: `type="tel"`

Met `<input type="tel">` markeert u een veld voor een telefoonnummer. Invoer wordt niet gevalideerd, omdat elke land nu eenmaal zijn eigen indeling voor telefoonnummers heeft. Het is daarom wel handig om zelf met `pattern` een patroon aan te geven gebaseerd op onze tiencijferige nummers met optioneel een streepje of spatie tussen net- en abonneenummer. Een voorbeeld is:

```
<input type="tel" pattern="0\d{1,3}[-|\s]\d{6,8} minlength="10" maxlength="11">
```

Dit patroon vereist dat het nummer met een 0 begint, waarna er 1 tot 3 cijfers volgen, daarna optioneel een streepje of een spatie, gevolgd door 6 tot 8 cijfers. De lengte is minimaal 10 en maximaal 11 tekens. Waterdicht is dit niet,

maar u kunt er de grootste missers mee voorkomen. Domweg het streepje of de spatie niet accepteren maakt geheel een stuk eenvoudiger.

```
<label for="tel">Telefoonnummer: <small>alleen cijfers</small></label>
<input id="tel" name="tel" type="tel" pattern="\d{9}" maxLength="10">
```

E-mailadressen: type="email"

Voor e-mailadressen gebruikt u `<input type="email">`. De ingebouwde validatie is eenvoudig gehouden, alleen een @-teken is vereist. Daardoor wordt zoets als `niet@compleet` geaccepteerd. Voor een striktere controle zult u `pattern` moeten gebruiken met een reguliere expressie voor e-mailadressen, bijvoorbeeld:

```
<label for="email">E-mailadres: <small>enandidonein.nl</small></label>
<input id="email" name="email" type="email"
      pattern="^([a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$)">
```

Met het attribuut `multiple` kunt u meerdere e-mailadressen laten invullen, die moeten worden gescheiden door een komma. De reguliere expressie uit het voorbeeld is daar overigens niet geschikt voor.

Webadressen: type="url"

Het type `url` maakt een veld voor de invoer van webadressen. De ingebouwde validatie controleert alleen of er een protocol is gebruikt, zoals `http://`, `https://` of `ftp://`. De grap is dat in feite elke lettercombinatie met deze opbouw door de controle komt, bijvoorbeeld `neus://`. Alleen het adresdeel is niet voldoende, dus `www.domein.nl` wordt niet geaccepteerd. Met het attribuut `pattern` kunt u afdwingen dat in elk geval een correct protocol wordt ingevoerd.

```
<label for="url">Webadres: <small>inclusief http://, https:// of ftp://</small></label>
<input id="url" name="url" type="url" pattern="^(http|https|ftp)://.+?">
```

Deze reguliere expressie kijkt alleen of de invoer begint met `http`, `https` of `ftp` gevolgd door `://`. De notatie `.+` betekent dat elk teken in elk aantal wordt geaccepteerd.

Wachtwoorden: type="password"

Verwacht geen versleuteling van het invoertype `password`, maar de invoer wordt wel verborgen door geen tekens maar bolletjes (of sterretjes) te tonen. Ook als het attribuut `value` is gebruikt, wordt die waarde als bolletjes in het veld getoond.

Omdat de inhoud van het veld niet wordt verstuurd, moet u een formulier met een wachtwoord nooit via GET maar via POST uuren (omdat de gegevens Verzendwijze: method). Het wachtwoord is dan als parameter tevoorschijn in de URL. Met POST bestaat het wachtwoord ook uit gewone tekst, maar is die (onzichtbaar) opgenomen in de body van de HTTP-uitvoerweg. Hierdoor is het ronduit onverstandig dergelijke informatie over een onveiligke verbinding te versturen; gebruik altijd HTTPS. Dit is echter geen bescherming voor een back over HTML.

Getallen: type="number"

Met het invoertype number maakt een veld voor de invoer van getallen. Als het veld de focus krijgt, worden in de browserknoppen kleine pijltjes getoond om de waarde te verhogen of verlagen. Om mogelijk apparaten dat een numeriek toetsenbord activeren. De grootte van de stap wordt bepaald door het attribuut step. De standaardwaarde is 1, maar elke waarde is mogelijk. Met value krijgt het veld een beginwaarde. De hoogste en laagste waarden kunnen worden ingesteld met min en max.



Decimalen met een punt

Gebruik voor decimalen een punt, niet de komma. Scheid niet de duizendtallen met een punt, zoals in 100.000. De Nederlandse notatie wordt door browsers niet gevuld, waardoor de punt als het Angelsaksische scheidingsteken voor decimalen wordt geïnterpreerd. Uw getal wordt dan dus duizend maal kleiner.

```
<label for="tafels">De tafel van 6</label>
<input id="tafels" name="tafels" type="number" value="0" step="6">
```

Het type number is niet geschikt voor elke numerieke invoer. Het moet zinnig zijn om de waarde met de pijlknoppen (de spinner) aan te passen. Daarmee is number ongeschikt voor waarden als het burgerservicenummer (BSN), een bankrekeningnummer (een IBAN bevat sowieso ook letters), een klantnummer enzovoort. In die situaties ligt het gebruik van type="text" meer voor de hand, al dan niet met een pattern.

Een bereik: type="range"

Met het type range wordt een schuifregelaar getoond waarmee een numeriek bereik wordt ingesteld. Het verschil met number is dat geen exacte waarde wordt ingevuld.

Standaard heeft range een bereik van 0 tot 100, met 50 als standaardwaarde. Met de attributen min, max en step kunt u dit aanpassen. De volgende code maakt een schuifregelaar die loopt van -100 tot 100, met 0 als standaardwaarde en stappen van 10:

```
<label> Form->bereik</label><br>
<input id="bereik" type="range" min="-100" max="100" step="10" value="0">
```

Aan het element is de ingestelde waarde niet te zien, en daar is alleen maar JavaScript wat aan te doen. Aan de voorgaande code wordt een `` toegevoegd waarin de waarde wordt getoond:

```
<label> Form->Slider</label>
<input id="Slider" type="range" min="-100" max="100" step="10" value="0">
<span class="waarde">0</span>
```

Met JavaScript wordt de waarde uitgelezen en in de span gezet:

```
<script>
  const slider = document.getElementById('Slider');
  const waarde = document.querySelector('.waarde');
  slider.addEventListener("input", () => {
    waarde.innerHTML = slider.value;
  });
  waarde.innerHTML = slider.value;
</script>
```

Analyse van de code:

- Het element `<input>` met de id `#Slider` wordt in de variabele `slider` opgeslagen.
- Het element `` met de klasse `.waarde` wordt in de variabele `warde` opgeslagen.
- Er wordt constant gecontroleerd of de waarde van `slider` verandert en als dat zo is wordt die waarde als tekst toegewezen aan de `` die is opgeslagen in de variabele `warde`.
- De vierde regel zorgt ervoor dat de beginwaarde van de schuifregelaar in de `` wordt getoond.
- De complete code is beschikbaar via [\[B6.html\]](#).

Einduren: type="color"

De gebruiker kan een kleur kiezen via `<input type="color">`. Zonder attributen wordt een vergelijkbaar gekozen. De waarde is de hexadecimale-kleurcode van RGB-kleur, zoals #FF0000. Misschien niet geactiveerd, dan wel.

de kleurkiezer van het systeem geopend. De daarin gekozen kleur wordt de waarde van dit element, wederom hexadecimaal. Het attribuut value stelt een andere standaardkleur in, bijvoorbeeld rood:

```
<label for="kleur">Kies een kleur:</label>
<input id="kleur" name="kleur" type="color" value="#ff0000">
```



Herhaling: RGB

RGB staat voor de kleuren rood, groen en blauw. Elke pixel kan voor elke kleur een waarde hebben van 0 tot 255. Een rode pixel heeft de waarden 255, 0, 0, een groene pixel 0, 255, 0 en een blauwe pixel 0, 0, 255. Hoe hoger de waarde, hoe groter de helderheid van de kleur. Zwart is 0, 0, 0 en wit is 255, 255, 255. Grijstinten ontstaan als de RGB-waarden gelijk zijn. Een middengrijze kleur is in RGB 128, 128, 128. Meer over kleur en de hexadecimale notatie in de hoofdstukken over CSS.

In een browsers die het invoertype color niet ondersteunt, ziet de gebruiker een tekstveld (de standaardwaarde van type). Om ervoor te zorgen dat het formulier dan toch met een geldige kleurwaarde wordt verstuurd, kunt u een pattern instellen. Voor een hexadecimale kleurwaarde is een reguliere expressie ^#[a-fA-F0-9]{6}. Hier staat: de tekenreeks begint met een #, gevolgd door zes kleine letters in het bereik a-f, hoofdletters A-F of cijfers van 0 tot 9. Wanneer u ook value instelt op een kleurwaarde en een title toevoegt, ziet de gebruiker wat de bedoeling is. De complete code wordt dan:

```
<label for="kleur">Kies een kleur:</label>
<input id="kleur" name="kleur" type="color" value="#ff0000" pattern="#[a-fA-F0-9]{6}" title="Een
waarde van #000000 tot #####">
```

Datum en tijd

Voor de invoer van datums en tijden zijn er verschillende typen.

- date laat een volledige datum kiezen: dag, maand, jaar;
- time een tijdkiezer in 24-uursindeling;
- datetime-local gecombineerde datum- en tijdkiezer;
- week een weeknummer;
- month een maand.

Het is de bedoeling dat de browser een element genereert waarin de gebruiker een datum of tijd kan kiezen in de indeling van zijn systeem, in Nederland dus dag-maand-jaar en uur:minuut (24-uursindeling). De gegevens worden

echter naar de server gestuurd in een internationaal gestandaardiseerde vorm, zodat programmeurs van de verwerkende systemen altijd precies weten welke indeling van datum en tijd ze kunnen verwachten. Die standaard is **YYYY-MM-DD** voor de datum en **UU:SS** voor de tijd. Bezoekers met een Nederlands systeem voeren datum en tijd dus in zoals ze gewend zijn, maar als u een script schrijft voor verwerking van deze gegevens gaat u uit van de internationale standaardnotatie.

Met `<input type="date">` kan de bezoeker een datum invoeren. Hoe precies hangt af van hoe de browserfabrikant het element heeft ingebouwd. Het attribuut `value` moet als u het voor de beginwaarde gebruikt een geldige datum zijn in de indeling **YYYY-MM-DD**, bijvoorbeeld `value="2020-02-02"` voor 2 februari 2020. De Nederlandse bezoeker krijgt dat te zien als **02-02-2020**. De attributen `min` en `max` zijn voor de oudste en nieuwste datum, ook in de indeling **YYYY-MM-DD**. Het attribuut `step` heeft standaard een waarde van 1 dag.

Een tijdinvoer bestaat uit `<input type="time">`. Ook hier hangt het uiterlijk van het element af van de browserfabrikant. De waarde moet bestaan uit twee cijfers voor de uren, een dubbelepunt en twee cijfers voor de seconden. Kwart voor vier 's middags is dus **15:45**.

Ook bij time kunnen de attributen `value`, `min`, `max` en `step` worden gebruikt. De standaardwaarde van `step` is 60 seconden.



Opmaak aanpassen is beperkt

Rij invoervelden die een besturingselement genereren, zoals een kleurkiezer of een datumkiezer, is het aanpassen van de opmaak met CSS vaak niet mogelijk. Tekstinvoervelden, knoppen, selectievakjes en keuzerondjes kunnen wel worden aangepast.

Selectievakjes: `type="checkbox"`

Een selectievakje wordt gemaakt met het attribuut `checkbox` en is bedoeld om een optie in of uit te schakelen ('aan- of uitvinken'). Kenmerkend is dat in een groep meerdere vakjes mogen zijn ingeschakeld. In de browser zien ze er standaard net zo uit als de selectievakjes in dialoogvenster van het besturingssysteem. Een ingeschakeld vakje is `true`, een uitgeschakeld `false`. Een vakje is standaard uitgeschakeld, maar kan bij het openen van het formulier worden ingeschakeld met het attribuut `checked`.

Bij het verzenden van het formulier worden alleen ingeschakelde vakjes meegenomen. Ze worden verstuurd als naam-waardepaar en daarom is het handig als u `value` instelt. Laat u de waarde weg, dan wordt een ingeschakeld vakje

verstuurd met de waarde on. Dat kan voldoende zijn bij een enkel vakje, maar bij een groep selectievakjes wordt dat lastiger (zie ook de paragraaf *Groeperen met <fieldset> en <legend>*).

```
<label for="checkbox">input type="checkbox"</label>
<fieldset id="checkbox">
<legend>Vergroot het comfort van uw e-bike met:</legend>
<label for="opt1">
<input id="opt1" type="checkbox" name="comfort" value="airco">airconditioning</label>
<label for="opt2">
<input id="opt2" type="checkbox" name="comfort" value="stuurverwarming">stuurverwarming</label>
<label for="opt3">
<input id="opt3" type="checkbox" name="comfort" value="zadelverwarming">zadelverwarming</label>
</fieldset>
```

The screenshot shows a user interface element where a legend is used to group three checkboxes. The legend text is "Vergroot het comfort van uw e-bike met:". Below it are three checkboxes labeled "airconditioning", "stuurverwarming", and "zadelverwarming".

Afbeelding 6.8 Selectievakjes gegroepeerd met *<fieldset>* en *<legend>* (06_03.html).

De opties zijn gegroepeerd met `name="comfort"`. Hier zou zonder `value` niet duidelijk zijn welke comfortopties zijn gekozen. Met een waarde voor `value` wordt een ingeschakeld vakje naar de server gestuurd als `comfort=airco` en `comfort=zadelverwarming`.

Keuzerondjes: `type="radio"`

Keuzerondjes lijken op selectievakjes, met een belangrijk verschil. In een groep mag maar één keuzerondje zijn ingeschakeld. De opties sluiten elkaar uit: als het A is, is het niet B of C. Een keuzerondje is het type `radio`. Losse keuzerondje komen niet voor, ze vormen altijd een groep. Een groep maakt u met de elementen `<fieldset>` en `<legend>` (net als bij selectievakjes; zie ook de paragraaf *Groeperen met <fieldset> en <legend>*). Een groep wordt gevormd door keuzerondjes met dezelfde naam. Het attribuut `value` geeft de gekozen optie aan. Om het aantal medewerkers in een bedrijf op te geven, codeert u bijvoorbeeld het volgende:

```
<label for="radio">input type="radio"</label>
<fieldset id="radio">
<legend>Aantal medewerkers</legend>
<label for="opt4"><input id="opt4" type="radio" name="nw" value="1">1 (alleen 1k)</label>
```

```
<label for="opt5"><input id="opt5" type="radio" name="nw" value="2-10">2 - 10</label>
<label for="opt6"><input id="opt6" type="radio" name="nw" value="11-50">11 - 50</label>
</fieldset>
```

input type="radio" Aantal medewerkers

- 1 (alleen ik)
- 2 - 10
- 11 - 50

Afbeelding 6.9 Van een groep keuzerondjes kan er altijd maar één zijn ingeschakeld (06_03.html).

Als de bezoeker het keuzerondje 2 tot 10 inschakelt, wordt die optie als `name=2-10` naar de server gestuurd. U ziet dat hier vooraf geen rondje is ingeschakeld. Dat kan wel, met het attribuut `checked`, maar het is vaak niet gewenst. Voor komt dat u een bezoeker onbedoeld beïnvloedt. Als de invoer niet vereist is (niet required) en de bezoeker maakt geen keus, dan komt het aantal medewerkers gewoon niet voor in het resultaat.

Is eenmaal een rondje geactiveerd, dan kan die groep niet meer leeg worden gemaakt. Wanneer een bezoeker de mogelijkheid moet hebben geen van de opties in de groep te selecteren, voegt u een extra keuze toe, bijvoorbeeld 'niet van toepassing' of 'geen voorkeur'. Het is nu eenmaal een kenmerk van keuzerondjes dat er één wordt geselecteerd. Als die keus optioneel is, is het keuzerondje misschien niet de beste optie. Overweeg een andere invoertype, bijvoorbeeld een keuzelijst met onder meer een neutrale optie zoals 'geen voorkeur'. Zie voor het maken van keuzelijsten de paragraaf *Kiezen uit een lijst: <select>, <datalist>, <option>*.

Een vraag waarop het antwoord ja of nee moet zijn is bij uitstek geschikt voor een keuzerondje. Met het attribuut `required` op een van de opties moet een van de opties in de groep worden gekozen.

```
<fieldset>
  <legend>Wilt u onze nieuwsbrief ontvangen?</legend>
  <label for="optin"><input id="optin" name="nwbrief" type="radio" value="ja" required>ja</label>
  <label for="optout"><input id="optout" name="nwbrief" type="radio" value="nee">nee</label>
</fieldset>
```

Bestanden uploaden: <type="file">

Om de gebruiker de mogelijkheid te bieden een of meer bestanden mee te sturen, gebruikt u <input type="file">. Dit genereert in elk geval een knop met het opschrift Bestand kiezen of vergelijkbaar. (U bepaalt de vormgeving met CSS.) De knop activeert het systeemvenster voor het openen van bestanden, waarin de gebruiker een keus kan maken.

Behalve het gebruikelijke attribuut name heeft dit invoertype ook de attributen accept en multiple. Zonder het attribuut accept wordt elk type bestand geaccepteerd. Met het attribuut en de waarden audio/*, video/* en image/* wordt de bestandskeuze beperkt tot een of meer gewenste typen. Meerdere typen worden van elkaar gescheiden met een komma. U mag ook een MIME-type aangeven en een bestandsextensies gebruiken – het W3C moedigt dat ook aan – maar als u meer wilt dan één bepaald type (bijvoorbeeld alleen docx-bestanden) wordt het al snel lastig, omdat er veel extensies bestaan en MIME-typen niet zijn gestandaardiseerd.



Geen beveiliging

Het attribuut accept biedt geen beveiling tegen verkeerde bestands-typen. Controleer bij de verwerking van geüploade bestanden altijd controleren of het juiste type is gebruikt en of niet met de extensie is gerommeld om een virus te verhullen.

Het attribuut multiple maakt het mogelijk meerdere bestanden te kiezen en te verzenden. Het opschrift van de knop kan daarbij in meervoud veranderen en het bijschrift kan aangeven hoeveel bestanden zijn geselecteerd, maar dit verschilt per browser. De bezoeker maakt op de gebruikelijke wijze een keus in het dialoogvenster. Dat wel zeggen met Ctrl ingedrukt individuele bestanden selecteren en met Shift een reeks.

Verzendknop: type="submit"

Voor het verzenden van een formulier gebruikt u een speciaal type dat een knop maakt: <input type="submit">. Het opschrift van de knop stelt u in met value. De standaardwaarde is Verzenden of een vergelijkbare tekst.

De knop heeft vergelijkbare attributen als het element <form>. Hiermee kunt u de in het formulier ingestelde waarden overschrijven. Dit biedt de mogelijkheid om verschillende verzendknoppen te maken met verschillende functies.

- **formaction** de URL van het bestand dat het formulier verwerkt;
- **formenctype** de codering van het formulier (zie enctype);
- **formmethod** de verzendmethode: GET of POST;
- **formnovalidate** de validatie van het formulier wordt uitgeschakeld.

Resetknop: type="reset"

Het paardenmiddel onder de invoertypen is de resetknop: `<input type="reset">`. Het opschrift van de knop stelt u in met het attribuut value. Het is alles of niets; het hele formulier wordt teruggebracht in de begintoestand.

Vraag u af of het zinvol is om een resetknop in het formulier op te nemen. Als de bezoeker deze per ongeluk activeert en zijn invoer kwijt is, bestaat de kans dat hij het formulier niet opnieuw invult en u een bestelling/aanvraag/reservering misloopt. Biedt alleen een resetknop aan als het echt nodig is en plaats die zo dat een onbedoelde reset wordt vermeden.

Afbeeldingknop: type="image"

Vergelijkbaar met de verzendknop is `<input type="image">`. Het grote verschil is dat geen knop met het opschrift Verzenden wordt getoond, maar een afbeelding. De functie blijft echter het verzenden van de formuliergegevens.

Het bijzondere is dat behalve de waarden van het formulier ook de x-y-coördinaten (in pixels) van het klikpunt in de afbeelding worden verstuurd (daarmee lijkt dit type enigszins op een imagemap). U krijgt de gegevens binnen als bijvoorbeeld `x=45&y=73`. Met het attribuut name geeft u meer betekenis aan deze waarden, bijvoorbeeld `locatie.x=45&locatie.y=73` (maar de browser moet dat ondersteunen).

```
<input type="image" src="afbeelding.png" alt='Speelwerelden' name="locatie">
```

Een toepassing kan zijn dat de bezoeker zijn locatie op een kaart aangeeft. Of u toont een afbeelding van werelden in een game waarmee de speler met een klik naar de gekozen wereld gaat. Let op, het blijft een verzendknop. Laat een gebruiker er niet halverwege een formulier zijn provincie mee kiezen, want dan wordt een half ingevuld formulier verzonden; gebruik voor visuele keuzen in een formulier een usemap of vergelijkbaar. (Een provincie kiezen is sowieso handiger met een keuzelijst).

Omdat het type een afbeelding is, zijn de attributen src (de bron) en alt (beschrijving) noodzakelijk. Het attribuut value moet niet worden gebruikt. Daarnaast heeft dit type dezelfde attributen als type="submit", aangevuld met width en height voor de breedte en hoogte van de afbeelding (maar die kunnen ook met CSS worden ingesteld).

Functieloze knop: type="button"

Er is ook het invoertype voor een algemene knop, `<input type="button">`. Deze knop heeft geen functie. Althans, niet zolang u er geen script voor schrijft. Het klikken op de knop wordt geregistreerd door de browser. Met JavaScript kunt u vervolgens bepalen welke actie na het klikken moet worden uitgevoerd.

Een alternatief voor `<input type="button">` is het element `<button>`, dat verderop wordt besproken.

Verborgen waarden: type="hidden"

Het is mogelijk om informatie met de server uit te wisselen die niet relevant is voor de gebruiker. Deze informatie kan onzichtbaar worden meegegeven via `<input type="hidden">`. Het attribuut `name` bevat de naam van het kenmerk en `value` de waarde. Meer attributen zijn er niet. Dit type is niet veiliger dan andere, zichtbare invoertypen. Ook de inhoud van `hidden` wordt als leesbare tekst verzonden.

In het algemeen hebt u scripting nodig om zinvolle verborgen informatie toe te voegen, bijvoorbeeld een functie die de datum en tijd van inzending aan het formulier toevoegt. Dit veld wordt ook vaak door internetaanbieders gebruikt om bijvoorbeeld (verborgen) de klantID en andere gegevens mee te sturen, zodat op de server kan worden gecontroleerd of degene die het formulierscript gebruikt ook daadwerkelijk klant is.

Het element `<button>`

In tegenstelling tot het invoerelement `<input type="button">` is het element `<button>` een echte knop. Dat biedt het voordeel dat de opmaak met CSS veel flexibeler is dan van het invoertype, terwijl `<button>` met het attribuut `type` en de waarden `submit` en `reset` gewoon fungereert als verzend- of herstelknop. Daarmee is deze knop beter in de opmaak van een formulier te integreren dan het invoertype button.

Zonder het attribuut `type` is button automatisch een verzendknop en beschikt u over de attributen `formaction`, `formenctype`, `formmethod`, `formnovalidate` en `formtarget`. Met `name` en `value` kan ook een naam-waardepaar worden meegestuurd.

Voor alle knoptypen zijn de attributen `autofocus`, `disabled` en `form` beschikbaar.

Bedenk dat een `<button>` geen standaardopschrift heeft, ook niet als verzend- of resetknop. De knop heeft sowieso geen inhoud, die moet u er zelf aan geven. Dat kan tekst zijn, zoals Verzend, maar bijvoorbeeld ook een afbeelding of een combinatie daarvan.

Voor een verzendknop waarmee u de webinterface van Outlook.com opent schrijft u de volgende code:

```
<form action="http://www.outlook.com">
  <button>Open het mailprogramma</button>
</form>
```



Neem geen link op in een knop

Merk op dat er in de knop niet een link met een URL wordt geplaatst, zoals `<button>...</button>`. Dat kan niet volgens de HTML-specificatie: een `<button>` mag geen interactieve elementen bevatten, zoals `<a>`. (Het omgekeerde kan ook niet; plaats geen `<button>` in `<a>`.) Buiten dat kunt u alle 'phrasing content' gebruiken en dat zijn heel veel elementen (zie hoofdstuk 2, *Het contentmodel*).

Kiezen uit een lijst: `<select>`, `<datalist>`, `<option>`

Om de bezoeker te helpen bij het kiezen van mogelijkheden kunt u opties opnemen in een keuzelijst. Een keuzelijst in een webpagina ziet er net zo uit als in een besturingssysteem en werkt ook op dezelfde manier. In het invoervak staat de huidige of voorgestelde optie en daarnaast staat een keuzepijl waarbij de gebruiker een lijst met mogelijkheden opent. De gebruiker klikt op een optie, de (nieuwe) keus wordt getoond in het tekstvak en wordt verder verwerkt. In een HTML-formulier betekent dat: de naam van de keuzemogelijkheid en de gekozen waarde worden verzonden. Dit type keuzelijst wordt gemaakt met het element `<select>`, een oudgediende in HTML.

In HTML 5 is een tweede soort lijst geïntroduceerd, ook wel bekend als keuzelijst met invoervak. Ook hiermee kan de bezoeker kiezen uit voorgebakken mogelijkheden, maar die verschijnen pas als hij begint te typen in het tekstvak en de lijst overeenkomstige waarden bevat. Staat de waarde die hij uiteindelijk invoert niet in de lijst, dan wordt de eigen invoer geaccepteerd. De bezoeker kan dus zelf bepalen wat hij invoert. De lijst is meer een hulpmiddel dan een keurslijf. Dit type lijst heet `<datalist>`.

Er is nog een verschil tussen de keuzelijsten: `<select>` kan zelfstandig in een formulier worden gebruikt, terwijl `<datalist>` een uitbreiding is op bijna alle

typen van het element <input>. Dit betekent dat u het invoerelement moet koppelen aan de datalist, waarover hierna meer.

<option>

Of u nu <select> of <input> met een <datalist> gebruikt, de keuzemogelijkheden staan altijd in een element <option>. Daarvan kunt u er zo veel gebruiken als u wilt. Als de lijst erg lang wordt of als er een logische scheiding tussen de opties bestaat, kunt u opties groeperen met het element <optgroup>.

<option> heeft naast de globale attributen ook:

- `disabled` het veld uitschakelen;
- `label` de voor de gebruiker zichtbare keuzemogelijkheid. Is te gebruiken in plaats van de inhoud van het element <option>;
- `selected` de optie is vooraf geselecteerd;
- `value` de waarde die naar de server wordt gestuurd.

<optgroup> heeft naast de globale attributen nog `disabled` en `label`. Het attribuut `label` is verplicht en bevat de naam van de groep.

In het volgende voorbeeld worden sluittags gebruikt, maar de sluittag is in vrijwel alle situaties optioneel. Omdat hier het element `inhoud` heeft is de sluittag voor de duidelijkheid toegevoegd. Ook bij <optgroup> is de sluittag optioneel.

```
<label for="gezond">De gezonde keus</label>
<select id="gezond" name="gezond">
  <optgroup label="groenten">
    <option value="spinazie">spinazie</option>
    <option value="witte kool">witte kool</option>
    <option value="tuinbonen">tuinbonen</option>
  </optgroup>
  <optgroup label="fruit">
    <option value="appel">appel</option>
    <option value="peer">peer</option>
    <option value="mandarijn">mandarijn</option>
  </optgroup>
</select>
```



Afbeelding 6.10 De opties zijn beschikbaar in de keuzelijst (06_03.html).

<select>

Algemene attributen van <select> zijn autofocus, disabled, name, required en form. Specifieke attributen zijn multiple voor meerkeuzemogelijkheden en size voor het aantal getoonde opties; de lijst is dan geheel of gedeeltelijk geopend. In het voorbeeld kan één mogelijkheid worden gekozen.

```
<label for="stijl">Kies één muziekstijl:</label>
<select id="stijl" name="stijl">
  <option value="1">pop</option>
  <option value="2">rock</option>
  <option value="3">indie</option>
  <option value="4">dance</option>
</select>
```

Moeten meer keuzen mogelijk zijn, dan ziet <select> uit het voorgaande voorbeeld er anders uit. De rest van de code blijft hetzelfde. De lijst wordt (deels) geopend en de gebruiker kan met Ctrl of Shift ingedrukt zijn selectie maken.

```
<label for="stijl11">Kies een of meer stijlen <small>(houd bij het selecteren Ctrl of Shift ingedrukt)</small></label>
<select id="stijl11" name="stijl11" multiple>
  <option value="1">pop</option>
  <option value="2">rock</option>
  <option value="3">indie</option>
  <option value="4">dance</option>
</select>
```

Deze manier van selecteren is niet gemakkelijk. Als meer opties mogen worden gekozen, kunnen beter selectievakjes worden gebruikt.

In plaats van een label kunt u een optie toevoegen met een lege waarde. Deze optie moet de eerste in de lijst zijn. Wanneer de gebruiker geen ander keuze maakt, wordt wel de naam stijl verstuurd, maar de waarde is leeg. Dat dit mogelijk is, betekent niet dat u het moet doen. Het probleem is zichtbaar zodra de keus is gemaakt; u kunt niet meer zien bij welke vraag het antwoord hoort.

```
<select name="stijl">
  <option value="">Kies één muziekstijl</option>
  <option value="1">pop</option>
  <option value="2">rock</option>
  <option value="3">indie</option>
  <option value="4">dance</option>
</select>
```

Het belangrijkste attribuut van het element `<option>` is `value`. Dit bevat de waarde die wordt verstuurd met de naam van `<select>`. Andere attributen zijn `disabled`, `label` en `selected`.

Met `selected` wordt vooraf een keuze ingesteld. In combinatie met `multiple` kunnen meerdere `<option>`-elementen het attribuut `selected` hebben.

<datalist>

De in HTML 5 geïntroduceerde keuzelijst met invoervak `<datalist>` gebruikt ook de `<option>`-elementen voor de keuzemogelijkheden. Het invoerelement is `<input>`. Het kan voor bijna alle invoertypen worden gebruikt: `text`, `tel`, `url`, `email`, `date`, `time`, `number`, `range` en `color`. Belangrijk is dat `<input>` het attribuut `list` heeft met de `id` van de `<datalist>`. Als we de keuzelijst met muziekstijlen ombouwen tot een catalijst, is de code:

```
<label for="genre-lijst">Welke muziekstijl wil je horen?</label>
<input list="genre" id="genre-lijst" name="stijl">
<datalist id="genre">
  <option value="pop">
  <option value="rock">
  <option value="indie">
  <option value="dance">
</datalist>
```

De bezoeker ziet nu geen keuzemogelijkheden. Wanneer hij in het invoervak typt en de letters komen overeen met de mogelijkheden in de `<option>`-elementen, verschijnen die opties en kan hij die selecteren. Wordt bijvoorbeeld `populair` getypt, dan verschijnt de optie `pop`, totdat `ulair` wordt ingevoerd, want dan is er geen overeenkomst meer.

Dit type datalijst biedt de gebruiker dus vrije invoer, maar met automatisch aanvullen zolang de invoer overeenkomt met een of meer opties.



Let op de id's

Als u experimenteert met een formulier met daarin verschillende varianten van keuzelijsten, let er dan op dat de gebruikte id's uniek zijn. Het resultaat wordt onvoorspelbaar als dat niet zo is; en het document is niet valide.

Nu ondersteunt (iOS) Safari 11st en `datalist` niet. Het kan daarom zinvol zijn een combinatie te maken met `<select>`. Op die manier hebt u een terugvalmechanisme en biedt elke browser de mogelijkheden uit de lijst aan, mits u de elementen in de hier getoonde volgorde gebruikt.

```
<label for="genre-lijst1">Welke muziekstijl wil je horen?</label>
<input list="genre" id="genre-lijst1" name="stijl">
<datalist id="genre">
  <label for="lijst-fall">Kies een muziekstijl:</label>
  <select id="lijst-fall" name="stijl">
    <option value="">
    <option value="pop">pop</option>
    <option value="rock">rock</option>
    <option value="indie">indie</option>
    <option value="dance">dance</option>
  </select>
</datalist>
```

Merk op dat het element `<option>` nu ook inhoud heeft. Dat is nodig omdat anders de lijst in oude browsers leeg blijft. Deze constructie doet het volgende:

- als `<datalist>` wordt ondersteund, is die via het attribuut `list` aan `<input>` gekoppeld en worden binnen `<datalist>` alle andere elementen dan `<option>` genegeerd (vet in de code);
- een browser die `<datalist>` niet ondersteunt, negeert dat element en gebruikt `<select>` en de `<option>`-elementen. De gebruiker ziet echter wel het invoerveld voor tekst en dat is minder elegant, maar met alleen HTML niet te vermijden.



Geen opmaak met CSS

Het is (nog) niet mogelijk om de opties in een `<datalist>` op te maken met CSS. U zit vast aan wat de browser ervan maakt. Opties als onderdeel van `<select>` zijn wel met CSS op te maken.

Uitgebreide tekstinvoer: <textarea>

Voor opmerkingen of aanvullende informatie bij een bestelling in uw webshop is één tekstregel misschien niet voldoende en dan kunt u uitwijken naar een ander element. Met <textarea> maakt u een vak voor tekstinvoer. Hierin kunnen meerdere regels tekst en ook meerdere alinea's tekst worden ingevoerd. De attributen cols en rows bepalen hoeveel tekens breed het vak wordt en hoeveel regels hoog, maar u kunt de afmetingen ook (beter) instellen met de CSS-eigenschappen width en height. Vooral de breedte hebt u beter onder controle met CSS. Op de hoeveelheid in te voeren tekst hebben cols en rows geen invloed. Als de tekst niet meer past, verschijnt er een schuifbalk. U kunt de hoeveelheid tekst beperken met het attribuut maxlength. Als de verwachte invoer een minimale lengte heeft, stelt u dat in met minlength. Gaat het u er alleen om dat het veld wordt ingevuld, gebruik dan required.

Met het attribuut wrap wordt ingesteld of regeleinden (returns) meekomen als het formulier wordt verzonden. Het attribuut heeft twee waarden: soft en hard. Met soft (standaardwaarde) komen alleen de regeleinden mee die de gebruiker met Enter invoert. Regeleinden die ontstaan doordat de rechterkant van het invoervak is bereikt, komen niet mee. Met hard komen alle regeleinden mee, ook die ontstaan doordat de tekst automatisch op de volgende tegel doorloopt. Bij de instelling hard is het attribuut cols verplicht.

Ook attributen zoals autofocus, placeholder en readonly zijn bruikbaar bij <textarea>.

```
<label for="mening">Opmerkingen bij uw bestelling:</label>
<textarea id="mening" name="mening" rows="5">
</textarea>
```

Groeperen met <fieldset> en <legend>

De elementen <fieldset> en <legend> kwamen al voorbij bij de keuzerondjes (<input type = "radio">) en selectievakjes (<input type = "checkbox">). Ze worden gebruikt voor het groeperen van invoerelementen die bij elkaar horen. De toepassing is niet beperkt tot de rondjes en vakjes. Elke logische eenheid van invoerelementen kan worden gegroepeerd.

<fieldset> maakt de groep. De bijbehorende invoerelementen moeten binnen de begin- en eindtag staan. Een titel of bipschrift voor de groep komt in het element <legend>. Dat kan gewoon tekst zijn, maar sinds HTML 5.2 ook een kop (<h1>-<h6>). Ook label en input kunnen in legend staan en daar

wordt. Wanneer u <legend> gebruikt, moet dat het eerste element na <fieldset> zijn. Velden voor een verzendadres en een factuuradres kunt u bijvoorbeeld zo coderen:

```
<fieldset>
  <legend>Verzendadres</legend>
  <label for="verzend_naam">Naam:</label>
  <input type="text" id="verzend_naam" name="verzend_naam">
  <label for="verzend_straat">Straat:</label>
  <input type="text" id="verzend_straat" name="verzend_straat">
  <label for="verzend_nummer">Huisnummer:</label>
  <input type="text" id="verzend_nummer" name="verzend_nummer">
  <label for="verzend_postcode">Postcode:</label>
  <input type="text" id="verzend_postcode" name="verzend_postcode">
  <label for="verzend_plaats">Plaats:</label>
  <input type="text" id="verzend_plaats" name="verzend_plaats">
</fieldset>

<fieldset>
  <legend>Factuuradres</legend>
  <label for="factuur_naam">Naam:</label>
  <input type="text" id="factuur_naam" name="factuur_naam">
  <label for="factuur_straat">Straat:</label>
  <input type="text" id="factuur_straat" name="factuur_straat">
  <label for="factuur_nummer">Huisnummer:</label>
  <input type="text" id="factuur_nummer" name="factuur_nummer">
  <label for="factuur_postcode">Postcode:</label>
  <input type="text" id="factuur_postcode" name="factuur_postcode">
  <label for="factuur_plaats">Plaats:</label>
  <input type="text" id="factuur_plaats" name="factuur_plaats">
</fieldset>
```

Ook op <fieldset> kan het attribuut `disabled` worden gebruikt. Daarmee worden alle elementen in die groep uitgeschakeld. Met `disabled` en een minimaal stukje JavaScript maakt u in een handomdraai een formulier dat direct reageert op keuzen van de gebruiker. In het volgende voorbeeld kan de gebruiker aangeven of het factuuradres afwijkt van het verzendadres. Zo ja, dan worden de velden voor het factuuradres vrijgegeven; standaard zijn die uitgeschakeld.

Aan het voorgaande voorbeeld voegen we voor de eerste <fieldset> wat HTML toe:

```
<fieldset>
<legend>
  <label for="factuurIsAnders">
    <input type="checkbox" id="factuurIsAnders" onchange="form.factuur.disabled = !checked"> Het
    factuuradres wijkt af van het verzendadres
  </label>
</legend>
```

De tweede `<fieldset>` krijgt het attribuut `name` met de waarde `factuur` en wordt uitgeschakeld:

```
<fieldset name="factuur" disabled>
```

Na de sluittag van de laatste `<fieldset>` wordt ook de toegevoegde `<field-`
`set>` afgesloten. (De complete code is te vinden in `06_03.html`.)

Nu gebeurt het volgende:

- 1 De factuur-`<fieldset>` kan met JavaScript worden aangesproken via de `name`.
- 2 De invoeropties zijn bij het laden van het formulier uitgeschakeld met `disabled`.
- 3 Een verandering van de toestand van het selectievakje wordt met `onchange` opgevangen.
- 4 Als de toestand verandert, wordt van het formulierelement met de naam "factuur" het attribuut `disabled` uitgeschakeld en kunnen de opties wor- den gekozen. In feite wordt `disabled` *omgeschakeld*; als het uit staat gaat het aan en als het aan staat gaat het uit.

Het factuuradres wijkt af van het verzendadres

Verzendadres	Factuuradres
Naam	Naam
Straat	Straat
Huisnummer	Huisnummer
Postcode	Postcode
Plaats:	Plaats:

Afbeelding 6.11 De velden voor het factuuradres kunnen alleen worden ingevuld als het selectievakje is ingeschakeld (`06_03.html`).

Het volgende voorbeeld laat zien dat de schakelaar ook binnen de uitgescha-
kelde `<fieldset>` kan staan:

```
<fieldset name="klantenkaart" disabled>
<legend><label>
```

```
<input type="checkbox" name="klant" onchange="form.klantenkaart.disabled = !checked">
Ik wil mijn klantenkaart gebruiken</label></legend>
<label for="klantnaam">Naam op de kaart:</label>
<input name="klantnaam" id="klantnaam">
<label for="klantnummer">Kaartnummer:</label>
<input name="klantnummer">
</fieldset>
```

Hoe kan het dat het eerste selectievakje wel beschikbaar is en de velden niet? Alles in de `<fieldset>` is toch uitgeschakeld? In de HTML-specificatie is vastgelegd dat in dit geval `disabled` geldt voor alle invoerelementen in `<fieldset>`, behalve de invoerelementen in `<legend>`. (Om precies te zijn, het eerste `<legend>`-kind van `<fieldset>`.)

`<fieldset>`-elementen mogen worden genest. Dat biedt de mogelijkheid om uitgebreide formulieren te maken waarbij u de beschikbaarheid van vervolgopties afhankelijk maakt van eerder gekozen opties. Zorg er daarbij voor dat u de keuzemogelijkheid steeds opneemt in een `<legend>`-element dat direct na de bijbehorende `<fieldset>` komt.

<input type="checkbox"/> Ik wil mijn klantenkaart gebruiken	
Naam op de kaart:	
Kaartnummer:	

Afbeelding 6.12 Velden in `<fieldset>` zijn uitgeschakeld, maar `<legend>` blijft beschikbaar (`06_03.html`).

Voortgang tonen: `<progress>`

HTML 5 bevat nog een aantal elementen waarmee u leuke dingen kunt doen. De keerzijde is dat u meestal een script nodig hebt voor een nuttige toepassing. Dat geldt ook voor `<progress>`, hoewel daar nog omheen valt te werken.

`<progress>` is een voortgangsindicator zoals u die kent van het installeren van software of het kopiëren van bestanden. Het element heeft twee attributen:

- `max` geeft aan bij welke waarde de taak is voltooid en mag een kommagetal bevatten. Als `max` wordt weggelaten, is de waarde standaard 1,0. (Let op: `max` mag ook 100 zijn als u van 0 tot 100% wilt gaan of 23,1 als die waarde beter past bij de taak.)
- `value` bevat de huidige voortgang en heeft een waarde die ligt tussen 0,0 en 1,0 of tussen 0,0 en de waarde van `max`.



Decimale punt

Gebruik een punt om decimalen aan te geven. Met de in Nederland gebruikelijke kommaannotatie werkt het niet.

<progress> kan een bepaalde of een onbepaalde waarde hebben. De waarde is bepaald als u het attribuut value gebruikt. Laat u dat weg, dan is de waarde onbepaald. Zo'n onbepaalde voortgangsindicator kunt u gebruiken als onbekend is hoe lang een taak zal duren. Aan de Een voortgangsindicator is te zien dat eraan wordt gewerkt en zodra de taak is voltooid wordt de indicator verborgen. Deze verpletterend eenvoudige HTML-code toont alleen een lopende indicator, zonder aanduiding van hoe ver het proces is gevorderd:

```
<progress></progress>
```

Een toepassing zonder JavaScript is bijvoorbeeld een formulier dat uit meerdere pagina's bestaat. Zijn er vier te doorlopen pagina's, dan is na elke pagina 25% meer voltooid en dat kunt u laten zien in een voortgangsbalk, die er dan achtereenvolgens zo uitziet:

```
<progress max="100" value="25">25</progress>
<progress max="100" value="50">50</progress>
<progress max="100" value="75">75</progress>
<progress max="100" value="100">100</progress>
```



Terugvalwaarde

Inhoud van <progress> wordt alleen getoond als een browser het element niet ondersteunt. Gebruik die daarom als terugvalwaarde, zoals in het voorgaande codevoorbeeld.

Schaal: <meter>

Het element <meter> is ook bedoeld om waarden te tonen, maar niet zoals de voortgangsindicator 'van nul tot klaar'. Bij meter kunt u denken aan de score bij een toets of een snelheidsmeter (downloadsgeschwindigheid), waarbij de onder- en bovengrens vastliggen. Dat laatste is belangrijk: het moet gaan om een begrensd bereik.

<meter> heeft de volgende attributen:

- value toont de huidige waarde (verplicht);
- min is de laagst mogelijke waarde;

- `max` is de hoogst mogelijke waarde (als `max` ontbreekt, is de waarde 1,0);
- `low` is de bovengrens van het lage bereik;
- `high` is de ondergrens van het hoge bereik;
- `optimum` is de optimale waarde.



Decimale punt

Gebruik een punt om decimalen aan te geven. Met de kommanotatie werkt het niet.

Om bij het voorbeeld van de toets te blijven; gewoonlijk is 1 het slechtste resultaat. Het maximum en optimum zijn beide 10 en vanaf 5,5 is het resultaat voldoende. Deze student heeft een 5,4 gescoord. In code is dat:

```
<meter value="5,4" min="1" low="5,5" high="8" max="10" optimum="10"> </meter>
```

Als u de `value` uit dit voorbeeld van laag naar hoog varieert en het resultaat in de browser bekijkt, ziet u dat de kleur van de balk verandert van rood en geel naar groen. Chrome is rood tot 5,5, geel tot 8 en groen vanaf 8; zoals u zou verwachten. In Firefox is 8 nog geel.



Terugvalwaarde

Inhoud van `<meter>` wordt alleen getoond als een browser het element niet ondersteunt. Dat gedrag is te gebruiken als terugvalmechanisme.

Resultaat: `<output>`

Het element `<output>` toont het resultaat van een berekening. Het is weliswaar een invoerelement en de waarde kan worden verstuurd, maar de gebruiker kan de waarde alleen indirect 'instellen', bijvoorbeeld door in ander velden bedragen in te voeren. Zonder JavaScript kunt u hier niets mee, want HTML heeft geen element `<telop>` of `<vermenigvuldig>`.

Het begrip 'berekening' mag u ruim zien, want bijvoorbeeld de waarde van een schuifregelaar (`<input type="range">`) kan ook met `<output>` worden getoond.

`<output>` heeft drie attributen:

- `for` geeft aan uit welke invoerelementen de uitvoer is samengesteld;
- `form` geeft aan bij welk formulier het element hoort;
- `name` is de naam die bij het verzenden wordt gebruikt.

Het voorbeeld werkt met twee invoervelden. De JavaScript-code is onvermijdelijk, maar wordt hierna toegelicht.

- De code is beschikbaar in 06_output.html.

```
<form oninput="o.value = parseInt(a.value) + parseInt(b.value)">
<input name="a" id="a" type="range" min="-100" max="100" value="0"> +
<input name="b" id="b" type="number" step="any" value="0"> =
<output name="o" for="a b">0</output>
</form>
```

Analyse van de code

De invoer in het formulier wordt opgevangen met de event `oninput`, dat de waarde (`value`) van het element met de naam "o" bepaalt door de waarde van het element met de naam "a" op te tellen bij de waarde van het element met de naam "b". De functie `parseInt()` zorgt ervoor dat de invoer als getallen wordt behandeld, zodat ermee kan worden gerekend. Dat is belangrijk, omdat de invoer als tekenreeks (`string`) binnenkomt en dan heeft het plussteken als functie de tekst achter elkaar te zetten; tekst kan niet worden opgeteld. (Als `string` geeft "11" + "22" het resultaat "1122", niet 33.)

Daarna worden de invoerelementen gedefinieerd. De `name` wordt gebruikt in de JavaScript-code van regel 1 (`a.value`). De `id` wordt gebruikt door het attribuut `for` van `output`.

Bij `<output>` komt alles samen. Met `o.value` uit de eerste 1 krijgt `<output name="o">` zijn waarde uit de optelling. `for="a b"` geeft aan dat de waarde afkomstig is van de elementen met de naam "a" en "b" (namen worden door een spatie gescheiden).

Samenvatting

Formulieren zijn een van de grotere onderwerpen van HTML. Het is ook een belangrijk onderdeel voor het contact met uw bezoekers. Goed formulierontwerp is niet eenvoudig en het begint met het kennen van de HTML-elementen. De hoofdpunten op een rij:

- Een formulier wordt gemaakt met `<form>`. Met attributen worden de verwerkingsinstructies bepaald en wordt ingesteld of de gegevens moeten worden gevalideerd.
- Een formulier wordt verwerkt op de webserver. Daarvoor gebruikt de webserver bijvoorbeeld PHP of een andere taal.

- Invoerelementen worden opgebouwd uit een element `<label>` en het invoerveld `<input>`.
- De kracht van formulier zit in het attribuut type van `<input>`. Daarmee kunnen veel soorter invoer worden ontvangen, zoals tekst, een e-mailadres of een telefoonnummer. Het voordeel daarvan is dat vooraf kan worden gecontroleerd of de invoer overeenkomt met wat er wordt verwacht. Op mobiele apparaten wordt daarbij automatisch het meest geschikte schermtouetsenbord geactiveerd.
- Invoer kan ook plaatsvinden via lijsten met vooraf ingestelde opties of een afbeelding met klikbare gebieden.
- Elementen zoals `<meter>` en `<progress>` zijn alleen nuttig in combinatie met scripting.

Oefeningen

- Om formulieren echt te kunnen verzenden, informeert u bij het bedrijf dat uw website host naar de mogelijkheden. Lukt dat niet, dan kunt u ook uitwijken naar een aanbieder als formspree.io of wufoo.com.
 - Hebt u geen mogelijkheid om het formulier te verzenden, laat dan voor de oefeningen de attributen `action` en `method` van `<form>` leeg.
- Maak een HTML-pagina met een eenvoudig contactformulier waarmee bezoekers op- en aanmerkingen kwijt kunnen.
 - Vraag naar naam, e-mailadres en opmerkingen.
 - Welke velden moeten per se worden ingevuld en hoe bereikt u dat?
- Maak een uitgebreider formulier met een korte enquête. Neem daarin vragen op met antwoorden zoals ja, geen mening, nee. Met welk element zou u dergelijke vragen markeren? Hoe markeert u vragen waarop meer dan een antwoord mogelijk is?
- Maak een bestelformulier voor enkele producten. Denk daarbij aan de mogelijkheid om van een product varianten te bestellen, bijvoorbeeld een bepaalde kleur of kledingmaat.



Gebruik de voorbeeldcode

Van verschillende afbeeldingen is de HTML- en CSS-code beschikbaar. De bestandsnaam staat in het bijschrift. Gebruik deze code om te oefenen: pas de code aan, voeg dingen toe, speel ermee! Kijk op handboek-html-css.nl.

Tabellen maken

Gestructureerde informatie komt het best tot zijn recht in een gestructureerde weergave. Overzichten, cijferlijsten, adres- en telefoonlijsten, het zijn maar een paar voorbeelden van gegevens die zich prima laten ordenen in tabellen. In HTML5 beschikt u over zo'n tien elementen en enkele attributen om gegevens te ordenen in rijen en kolommen.

U leert in dit hoofdstuk:

Hoe een tabel in HTML is opgebouwd.

Het toevoegen van een bijschrift en kop- en voetteksten.

Het maken van tabelcellen met rijen en kolommen.

Rijen en kolommen meerder cellen laten overspannen.

Inleiding

Met de komst van HTMLS kon de tabel weer worden gebruikt waarvoor hij is bedoeld: het weergeven van gestructureerde informatie. Mocht u net instappen: tabellen hebben jarenlang dienstgedaan als lay-outhulpmiddel, of beter: als het fundament, de wanden en het dak van websites. Met het volwassen worden van CSS en de groeiende mogelijkheden voor het maken van mooie, solide en flexibele lay-outs kan de tabel terug naar zijn oorsprong: *geordende lijst, voor het leesbaar maken van een (groot) aantal feiten of gegevens, gewoonlijk enkel in namen en cijfers en zo gegroepeerd, dat men ze gemakkelijk kan overzien* (Dikke Van Dale).

Tabellen kent u dus. In rijen en kolommen die elkaar kruisen en daarmee een raster van cellen vormen, wordt informatie geplaatst. Cijfers, tekst, afbeeldingen, hyperlinks, het kan allemaal in een tabel worden opgenomen (er gingen immers ooit complete webpagina's in). Tabellen kunnen bijzonder complex zijn, met cellen die meerdere kolommen of meerdere rijen overspannen. In die zin zijn tabellen in HTML niet anders dan spreadsheets of tabellen in kantoorsoftware. Het verschil zit natuurlijk in de manier waarop u een tabel maakt. Daarover gaat dit hoofdstuk. U leert alles over `<table>`, `<tr>`, `<td>` en al die andere elementen die van uw data een keurige tabel maken.

De structuur van een HTML-tabel

Een tabel kan nog verrassend veel verschillende elementen bevatten, als u alle mogelijkheden benut. Lang niet alle elementen zijn verplicht en diverse sluittags mogen worden weggelaten, waardoor een tabel zo eenvoudig kan zijn als het volgende voorbeeld van een tabel met één rij en één kolom:

```
<table>
  <tr><td>Tabel met 1 cel
</table>
```

Voor een beter overzicht adviseren we echter in een tabel ook alle sluittags te gebruiken. Het kan bovendien helpen om de elementen op afzonderlijke ingsprongen regels te typen. De aanbevolen notatie ziet er dan zo uit:

```
<table>
  <tr>
    <td>Tabel met 1 cel</td>
  </tr>
</table>
```

Zo eenvoudig is een tabel natuurlijk nooit. We bekijken de opbouw van een tabel aan de hand van iets realistischer voorbeeld: een overzicht van HTML-versies:

```
<table>
  <tr>
    <td>1995</td>
    <td>1997</td>
    <td>1998</td>
    <td>1999</td>
    <td>2014</td>
    <td>2016</td>
    <td>2018 ?</td>
  </tr>
  <tr>
    <td>HTML 2.0</td>
    <td>HTML 3.2</td>
    <td>HTML 4.0</td>
    <td>HTML 4.01</td>
    <td>HTML5</td>
    <td>HTML5.1</td>
    <td>HTML5.2</td>
  </tr>
</table>
```

U ziet in deze HTML-code drie belangrijke tabelelementen: `<table>`, `<tr>` en `<td>`. In de afbeelding is te zien dat dit overzicht nog niet erg duidelijk is, doordat een rand of andere visuele aanwijzingen ontbreken. Ook staat de inhoud nogal dicht op elkaar. Maar u weet intussen: dat is opmaak en hier gaat het over structuur.

1995	1997	1998	1999	2014	2016	2018	2020
HTML 2.0	HTML 3.2	HTML 4.0	HTML 4.01	HTML 5	HTML 5.1	HTML 5.2	HTML 5.3

Afbeelding 7.1 Een tabel van twee rijen en twee kolommen, maar erg overzichtelijk is het nog niet (`07_01.html`).

De basis: <table>

De basis van elke tabel wordt gevormd door het element <table>. Het is de ancestor (voorouder) van alle andere tabelelementen. <table> heeft alleen de globale attributen.

In een tabel kunnen nogal wat andere elementen worden gebruikt, die echter lang niet allemaal verplicht of noodzakelijk zijn. Als u ze gebruikt, is de volgorde wel belangrijk. De juiste volgorde is:

- <table>
- <caption>
- <colgroup>
- <thead>
- <tfoot>, en die komt voor of achter
 - nul of meer <tbody>
 - een of meer <tr>
- De cellen worden gemarkerd met <th> en <td>.



Geen sluittags

In tabellen mag u bij de elementen <colgroup>, <tbody>, <thead>, <tfoot>, <tr>, <th> en <td> de sluittag weglaten, hoewel het geen lege elementen zijn (het enige lege element is <col>). We adviseren u die tags voor de overzichtelijkheid wel te gebruiken.

Rijen en kolommen: <tr> en <td>

In de voorbeeldtabel zijn er twee rijen en acht kolommen en daarmee zijn zestien tabelcellen gemaakt. Cellen bevinden zich op het snijpunt van rijen en kolommen. Voor de tabelgegevens hebt u daarom allereerst een rij nodig. Een rij wordt gemaakt met het element <tr> (table row). In die rij komen de kolommen (cellen) en die bestaan uit het element <td> (table data) of <th> als het om koprijen gaat (zie ook de paragraaf over <thead>).

De elementen <td> en <th> hebben twee belangrijke attributen voor het indelen van de tabel: colspan en rowspan. Met colspan wordt aangegeven dat de cel meerdere kolommen overspant en met rowspan meerdere rijen. Zodra u deze attributen gebruikt, helpt het enorm om (desnoods tijdelijk) zichtbare randen te gebruiken om de tabelstructuur overzichtelijk te houden. Ook het schetsen van de lay-out voordat u begint te coderen is een handig hulpmiddel. In de volgende paragraaf wordt colspan gebruikt voor een koprij.

Koptekst:<thead>

Het element <thead> is vergelijkbaar met de header van een pagina of artikel: het groepeert tabelelementen die bij de kop van de tabel horen. Gebruik is niet verplicht. De rijen worden gewoon aangegeven met <tr> en voor de inhoud van een kopcel is er het element <th>, maar in de tabelkop mogen ook 'gewone' cellen (<td>) worden gebruikt, als subkop bijvoorbeeld.



Een koprij is geen bijschrift

In een tabel zijn de koprijen bedoeld voor uitleg van de informatie in de kolommen. Voor een algemeen bijschrift, dat betrekking heeft op de hele tabel, is er het tabelelement <caption>, maar het element <figcaption> kan ook heel goed worden gebruikt. Dat moet dan wel samen met <figure>, maar een tabel is (bijna) altijd een kandidaat voor <figure>.

We kunnen de HTML-tabel uitbreiden met een koprij en twee kopteksten. De eerste koptekst overspant de eerste vier kolommen en de tweede de laatste vier kolommen. In de afbeelding is voor de leesbaarheid met CSS wat ruimte gemaakt (padding) en de tekst is gecentreerd.

```
<table>
  <thead>
    <tr>
      <th colspan="4">Prehistorie</th>
      <th colspan="4">Moderne tijd</th>
    </tr>
  </thead>
  <tr>
    <td>1995</td>
    <td>1997</td>
    <td>1998</td>
    <td>1999</td>
    <td>2014</td>
    <td>2016</td>
    <td>2018</td>
    <td>2020</td>
  </tr>
  <tr>
    <td>HTML 2.0</td>
    <td>HTML 3.2</td>
    <td>HTML 4.0</td>
```

```

<td>HTML 4.01</td>
<td>HTML 5</td>
<td>HTML 5.1</td>
<td>HTML 5.2</td>
<td>HTML 5.3</td>
</tr>
</table>

```

Prehistorie				Moderne tijd			
1995	1997	1998	1999	2014	2016	2018	2020
HTML 2.0	HTML 3.2	HTML 4.0	HTML 4.01	HTML 5	HTML 5.1	HTML 5.2	HTML 5.3

Afbeelding 7.2 Een tabel met header en daarin twee koppen die ieder vier kolommen overspannen. De stylesheet van de browser maakt de koppen vet en gecentreerd. De rand is met CSS gemaakt (07_02.html).

Een bijschrift: <caption>

Het elementen <caption> markeert het bijschrift van een tabel. Het element is niet verplicht. Figuurbijdrachten staan vaak onder de afbeelding, maar omdat tabellen behoorlijk lang kunnen zijn, is het goed gebruik het bijschrift boven de tabel te plaatsen. Dan is direct duidelijk wat er in de tabel te vinden is. Deze best practice komt terug in de specificatie, doordat <caption> direct na het element <table> moet komen.

Voorzien van bijschrift ziet de tabel er zo uit:

```

<table>
  <caption>
    <p>Tabel 1. Tijdslijn van HTML-specificaties.</p>
  </caption>
  <thead>
    <tr>
      <th colspan="4">Prehistorie</th>
      <th colspan="4">Moderne tijd</th>
    </tr>
  </thead>
  <tr>
    <td>1995</td>
    <td>1997</td>
    <td>1998</td>
    <td>1999</td>
    <td>2014</td>
  </tr>

```

```

<td>2016</td>
<td>2018</td>
<td>2020</td>
</tr>
<tr>
    <td>HTML 2.0</td>
    <td>HTML 3.2</td>
    <td>HTML 4.0</td>
    <td>HTML 4.01</td>
    <td>HTML 5</td>
    <td>HTML 5.1</td>
    <td>HTML 5.2</td>
    <td>HTML 5.3</td>
</tr>
</table>

```

Tabel 1. Tijdslijn van HTML-specificaties.

Prehistorie				Moderne tijd			
1995	1997	1998	1999	2014	2016	2018	2020
HTML 2.0	HTML 3.2	HTML 4.0	HTML 4.01	HTML 5	HTML 5.1	HTML 5.2	HTML 5.3

Afbeelding 7.3 Tabel met bijschrift: De centrering van het bijschrift komt uit de browserstylesheet (07_03.html).

Overige elementen

Met de hiervoor genoemde elementen kunt u prima tabellen maken. Wilt u de mogelijkheden nog wat uitbreiden, dan zijn daarvoor nog de elementen in de volgende paragrafen beschikbaar.

De tabelinhoud: <tbody>

De daadwerkelijke inhoud van de tabel kan in het element <tbody> worden geplaatst. Verplicht is dat niet, maar het is wel een duidelijke markering van de tabelinhoud. Een tabel kan meerdere <tbody>-elementen hebben. In een tabel met specificaties van een computer zouden verschillende rubrieken (basiscomponenten, extra voorzieningen, randapparatuur) in afzonderlijke <tbody>-elementen kunnen staan.

Voettekst: <tfoot>

Een overzicht, samenvatting of totaal van de kolom(men) wordt gemarkeerd met <tfoot>. In een kostenoverzicht kan dat de som van de kolom zijn, maar

bijvoorbeeld ook een rood of groen vinkje als oordeel op basis van de informatie in de kolommen. Het element `<tfoot>` mag ook onder aan de tabel staan, zolang u maar één `<tfoot>` in de tabel gebruikt.

Of de tabelvoettekst boven of onder in de code staat maakt voor de presentatie overigens niet uit: de browser toont de inhoud van `tfoot` altijd als laatste rij van de tabel. `<tfoot>` is niet verplicht.

De cellen in `<tfoot>` worden gemaakt met `<tr>` en `<td>`. Merk op dat de eerste cel een kop heeft die geldt voor de rij (`scope="row"`).

```
<tfoot>
  <tr>
    <th scope="row">Totaal</th>
    <td>€ 41,70</td>
    <td>€ 18,63</td>
    <td>€ 65,00</td>
  </tr>
</tfoot>
```

Kolomgroepen: `<colgroup>` en `<col>`

Het eveneens niet verplichte element `<colgroup>` kan worden gebruikt om de structuur van de tabel aan te geven. Het element is bruikbaar voor opmaak met CSS, omdat zo een hele groep kolommen ineens kan worden opgemaakt. (Dat is overigens beperkt tot de CSS-eigenschappen border, background, width en visibility.) We hebben dat gedaan met de twee groepen van onze tijdlijntabel:

```
<table>
  <caption>
    <p>Tabel 1. Tijdlijn van HTML-specificaties.</p>
  </caption>
  <colgroup>
    <col span="4">
    <col span="4">
  </colgroup>
  <thead>
    ...

```

`<colgroup>` groepeert kolommen met het element `span` of met kindelementen `<col>`. Elke `col` bestrijkt één kolom, maar kan meerdere kolommen overspannen met `span`. In onze tabel zouden de groepen ook kunnen worden gedefinieerd als:

```
<colgroup span="4"></colgroup>
<colgroup span="4"></colgroup>
```

Tabel 1. Tijdslijn van HTML-specificaties.

Prehistorie				Moderne tijd			
1995	1997	1998	1999	2014	2016	2018	2020
HTML 2.0	HTML 3.2	HTML 4.0	HTML 4.01	HTML 5	HTML 5.1	HTML 5.2	HTML 5.3

Afbeelding 7.4 Kolommengroepen gemarkeerd met een achtergrondkleur (07_04.html).

Voorbeelden van tabellen

Met de kennis uit de voorgaande paragrafen kan een tabel worden gemaakt. Begin met het element `<table>`, voeg al dan niet de optionele elementen `<caption>`, `<colgroup>`, `<thead>`, `<tfoot>` en `<tbody>` toe, en maak rijen met `<tr>`, kopcellen met `<th>` en datacellen met `<td>`.

Eerst een eenvoudig voorbeeld met een veelvoorkomende situatie, een lege hoekcel: de volgende tabel toont kenmerken van diverse producten, waarbij de cel linksboven leeg is. Merk op dat de lege cel (vet) geen sluittag heeft. Ook dat is een kwestie van voorkeur, maar een lege cel is zo goed herkenbaar.

```
<table>
  <tr>
    <th>
      <th>Product 1</th>
      <th>Product 2</th>
      <th>Product 3</th>
    </th>
  <tr>
    <th>Kenmerk 1</th>
    <td>ja</td>
    <td>ja</td>
    <td>nee</td>
  </tr>
  <tr>
    <th>Kenmerk 2</th>
    <td>ja</td>
    <td>ja</td>
    <td>ja</td>
  </tr>
  <tr>
```

```
<th>Kenmerk 3</th>
<td>nee</td>
<td>nee</td>
<td>ja</td>
</tr>
</table>
```

	Product 1	Product 2	Product 3
Kenmerk 1	✓	✓	✗
Kenmerk 2	✓	✗	✓
Kenmerk 3	✗	✗	✓

Afbeelding 7.5 Tabel met een lege hoekcel (07_05.html).



Lege cellen

Zo eenvoudig is het dus: voor een lege cel maakt u wel de cel, maar u geeft hem gewoon geen inhoud. Soms wordt een spatie toegevoegd (met de code), maar die is niet verplicht en moderne browsers bouwen de tabel zonder die spatie ook goed op.

Het volgende voorbeeld is een tabel met vijf kolommen en negen rijen, met alles erop en eraan. Het is nogal een lap code, maar met deze ruime notatie zijn de onderdelen het best herkenbaar.

```
<table>
  <caption>
    <span>Tabel 1</span> voorbeeld met alle tabelelementen en het overspannen van rijen en kolommen.
  </caption>
  <thead>
    <tr>
      <th colspan="4">De kop van de tabel</th>
      <th>Laatste kolom</th>
    </tr>
    <tr>
      <td colspan="4">Onderkop</td>
      <td>Onderkop</td>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <td>totaal 1</td>
      <td>totaal 2</td>
```

```
<td>totaal 3</td>
<td>totaal 4</td>
<td>totaal 5</td>
</tr>
</tfoot>
<tbody>
<tr>
    <th colspan="4">Tussenkop over 4 kolommen</th>
    <th>tussenkop</th>
</tr>
<tr>
    <td>kolom 1</td>
    <td>kolom 2</td>
    <td>kolom 3</td>
    <td>kolom 4</td>
    <td rowspan="2">kolom 5, 2 rijen</td>
</tr>
<tr>
    <td>kolom 1</td>
    <td>kolom 2</td>
    <td>kolom 3</td>
    <td>kolom 4</td>
</tr>
</tbody>
<tbody>
<tr>
    <th colspan="4">tussenkop over 4 kolommen</th>
    <th>tussenkop</th>
</tr>
<tr>
    <td>kolom 1</td>
    <td>kolom 2</td>
    <td>kolom 3</td>
    <td>kolom 4</td>
    <td rowspan="2">kolom 5, 2 rijen</td>
</tr>
<tr>
    <td>kolom 1</td>
    <td>kolom 2</td>
    <td>kolom 3</td>
    <td>kolom 4</td>
</tr>
</tbody>
</table>
```

Tabel 1 Voorbeeld met alle tabelelementen en het overspannen van rijen en kolommen.

De kop van de tabel				Laatste kolom
Onderkop				Onderkop
Tussenkop over 4 kolommen				Tussenkop
kolom 1	kolom 2	kolom 3	kolom 4	kolom 5, 2 rijen
kolom 1	kolom 2	kolom 3	kolom 4	
Tussenkop over 4 kolommen				Tussenkop
kolom 1	kolom 2	kolom 3	kolom 4	kolom 5, 2 rijen
kolom 1	kolom 2	kolom 3	kolom 4	
totaal 1	totaal 2	totaal 3	totaal 4	totaal 5

Afbeelding 7.6 Gecompliceerde tabel met rij- en kolomoverspanning ([07_06.html](#)).

Het tabelnummer staat hier in een ``-element, want dan kan het anders worden opgemaakt (met CSS) dan de bijschrifttekst. De kop staat in een `<thead>` en de rij met de totalen in een `<tfoot>`. De tabel is in twee logische blokken gesplitst met `<tbody>`. Zoals hiervoor al is opgemerkt: het gebruik van `<tbody>` is niet verplicht. Elke `<tbody>` begint hier met een tussenkop: `<th>`. De eerste overspant vier kolommen: `colspan="4"`. Er zijn daarom maar twee cellen en dus twee `<th>`-elementen nodig. In de vijfde kolom overspant de laatste cel twee rijen: `rowspan="2"`. Let erop dat daarom de volgende rij geen cel voor kolom 5 bevat.

Samenvatting

Alles wat u nodig hebt om tabellen te markeren is in dit hoofdstuk aan bod gekomen:

- De basis van een tabel bestaat uit de elementen `<table>` `<tr>` en `<td>`.
- Een bijschrift komt direct na `<table>` in het element `<caption>`.
- Een tabel kan heel goed een `<figure>` zijn en dan kan het bijschrift in `<figcaption>`.
- Net als artikelen kunnen tabellen headers (kopregels) en footers (toelichting) hebben.
- De body van een tabel kan in `<tbody>` worden opgenomen of worden verdeeld over meerdere elementen `<tbody>`.

- Rijen en kolommen worden overspannen met de attributen rowspan en colspan.
- Het is mogelijk om kolommen te groeperen met `<colgroup>` en `<col>`.

Oefeningen

Maak een HTML-pagina met een tabel, bijvoorbeeld met adresgegevens of productgegevens, zoals in dit hoofdstuk werd getoond.

- Gebruik de daarvoor bestemde elementen voor kolomkoppen en rijkoppen.
- Maak een bijschrift en een footer met toelichting. Bedenk dat bij verwijzingen vanuit de cellen naar de toelichting een element zoals `<sup>` handig kan zijn, eventueel met link naar toelichting die u dan als bladwijzer markeert.
- Oefen het gebruik van rowspan en colspan om rij- en kolomoverspanning in de vingers te krijgen. Het helpt om vooraf een schets te maken van de beoogde tabel.



Gebruik de voorbeeldcode

Van vrijwel alle afbeeldingen is de HTML- en CSS-code beschikbaar. De bestandsnaam staat in het bijschrift. Gebruik deze code om te oefenen: pas de code aan, voeg dingen toe, speel ermee! Kijk op handboek-html-css.nl.

De basis van CSS

In de voorgaande hoofdstukken is uitgelegd hoe de inhoud van een webpagina op een betekenisvolle manier wordt gemarkeerd met de elementen van HTML. Vanaf nu gaat het over de vormgeving. Die heeft zijn eigen taal: Cascading Style Sheets of kortweg CSS. Alles wat met opmaak te maken heeft, of dat nu de pagina-indeling, een lettertype, een kleur, een rand of een achtergrondafbeelding is, wordt ingesteld met CSS. Zelfs de witruimte op een pagina behoort tot het domein van CSS. De opmaaktaal is onlosmakelijk verbonden met HTML en kennis ervan is onmisbare bagage voor elke webontwerper. In dit hoofdstuk wordt de basis gelegd voor het werken met CSS.

U leert in dit hoofdstuk:

Wat CSS en CSS3 zijn en waarom CSS zo geweldig is.

Hoe de taal CSS in elkaar zit.

Hoe de browser de opmaakinstructies verwerkt.

Wat de cascade is en waarom stijlregels soms niet doen wat u verwacht.

Waar de CSS-instructies kunnen staan en hoe een CSS-bestand in elkaar zit.

Elementen benaderen met selectors.

Wat CSS is

In de voorgaande hoofdstukken hebt u allerlei voorbeelden van HTML in de browser kunnen bekijken, maar heel opwindend ziet het er allemaal niet uit. Daar komt verandering in als u zelf de opmaak gaat bepalen met CSS. Maar wat is CSS?

CSS is de afkorting van Cascading Style Sheets, in het Nederlands ook wel trapsgewijs opmaakmodel genoemd. Het begrip cascading/trapsgewijs is uiteraard een belangrijke eigenschap en die wordt hierna uitgelegd. CSS is de opmaaktaal voor HTML (en XHTML, XML). Het definieert welk lettertype wordt gebruikt, in welke grootte en met welke kleur, of er een achtergrondkleur of achtergrondafbeelding is, hoeveel ruimte er tussen elementen moet zijn enzovoort. Alles (of bijna alles) wat u aan opmaak kunt bedenken, kan met CSS worden gemaakt. Net als HTML wordt CSS beheerd door het World Wide Web Consortium (W3C, zie ook hoofdstuk 1).

Voorgeschiedenis

Als u vandaag begint met het vormgeven van webpagina's is CSS de vanzelf-sprekende techniek, maar dat is niet altijd zo geweest. Opmaak is lang verzorgd met opmaakelementen en opmaakattributen van HTML, die daaraan zijn toegevoegd wegens gebrek aan beter. Toch begint de geschiedenis van CSS al in 1994, een eeuwigheid geleden in deze sector. De eerste aanbeveling kwam in 1996 – CSS level 1 – in 1998 gevolgd door CSS level 2. Pas in 2000 was er één browser (Internet Explorer 5 voor de Mac) die 99% van CSS 1 ondersteunde.

Er begint een verwarringe periode waarin de bekendste versie van CSS wordt ontwikkeld: CSS 2.1. Het duurt eindeloos voor deze het stadium van aanbeveling bereikt, pas in 2011 is het zover. Dat betekent niet dat CSS in de jaren daarvoor niet werd gebruikt. Het probleem was alleen dat door het ontbreken van een standaard elke browserfabrikant zijn eigen gang ging, deels op basis van het concept van CSS, deels naar eigen inzicht. Dat maakte het effect van CSS-kenmerken onvoorspelbaar, met (grote) verschillen tussen de browsers. Ontwerpers bedachten noodgedwongen de ene hack na de andere om de problemen te omzeilen.

Sinds 2011 gaat het dankzij het vaststellen van de standaard (de W3C-aanbeveling *CSS Level 2 Revision 1*) steeds beter. Het is onder browserfabrikanten ook niet meer gebruikelijk om met de bijzonderste zelfbedachte features te komen, zoals lang wel het geval was. Nu is de norm: goede ondersteuning van de standaard.

En toen was er CSS3

CSS3 is niet uit de lucht komen vallen (de eerste concepten dateren van 1999!) en eigenlijk bestaat het niet. Althans, het is niet één ding. Alle voorgaande CSS-versies bestonden uit één groot document, maar CSS3 is opgebouwd uit modules. Er zijn modules voor kleur, selectors, achtergronden en randen, tekst, enzovoort. Het grote voordeel is dat de ontwikkeling veel sneller gaat. Elke module wordt in zijn eigen tempo ontwikkeld en kan als standaard worden ingevoerd zodra die klaar is. Er hoeft niet zoals vroeger te worden gewacht tot alle andere onderdelen ook klaar zijn. Heeft een module het tot standaard geschopt, dan hoeft het daarna ook niet afgelopen te zijn. Terwijl er al een standaard is waarmee iedereen kan werken, kan als dat nodig is de ontwikkeling van de module doorgaan. Een volgende standaard is dan niet van niveau 1, maar van niveau 2, niveau 3 enzovoort. Een voorbeeld is de module *CSS Backgrounds and borders Level 3* die een W3C-aanbeveling wordt en waarvan tegelijkertijd *Level 4* al in de steigers staat.



Volg de ontwikkeling

Op www.w3.org/Style/CSS/current-work kunt u precies zien welke status de modules hebben en welke mogelijkheden u (op korte of lange termijn) kunt verwachten.

Terwijl nieuwe CSS-onderdelen worden ontwikkeld, zitten de browserfabrikanten niet stil. Die experimenteren al met de mogelijkheden die nog in ontwikkeling zijn en doordat browsers vaak worden geüpdatet, zijn nieuwe mogelijkheden snel beschikbaar. Dat heeft naast voordeelen (u kunt al snel gebruikmaken van eigenschappen die nog niet zijn uitontwikkeld) ook nadelen. De uiteindelijke versie van de eigenschap kan net iets anders werken, u hebt soms nog een *vendor prefix* nodig om de optie aan de praat te krijgen en u zult een oplossing moeten bedenken voor browsers die de optie niet herkennen. (Een vendor prefix is een voorvoegsel bij de naam van CSS-eigenschap, bijvoorbeeld `-webkit-` of `-moz-`. Dit komt in een volgend hoofdstuk nog aan de orde.) Per saldo is de modulaire opbouw een enorme verbetering, omdat nieuwe mogelijkheden sneller worden ontwikkeld en sneller stabiele browserondersteuning hebben.

CSS3 is gewoon CSS

Zoals gezegd is CSS3 niet uit de lucht komen vallen. In de modules wordt voortgeborduur op CSS 2.1. Daardoor zullen webpagina's die zijn vormgegeven met 'oude' CSS blijven werken. Het CSS3-deel bestaat uit het toevoegen van nieuwe eigenschappen of nieuwe waarden voor bestaande eigenschappen. In die zin kunt u CSS3 zien als een synoniem voor 'alle mogelijkheden die zijn ontwikkeld sinds CSS 2.1'.

In het vervolg van dit boek gaat het gewoon over CSS. CSS3 is een handige naam om duidelijk te maken dat er een nieuw CSS-tijdperk is begonnen, maar in de praktijk doet het er niet toe of een module versie 1, 2, 3 of 6 is. Veel belangrijker is of de CSS die u wilt gebruiken door browsers wordt ondersteund. Dat zal dan ook uw uitdaging zijn bij het ontwikkelen van websites en apps: met welke CSS-mogelijkheden kunt u een pagina-ontwerp het best uitwerken en hebben die mogelijkheden voldoende browserondersteuning?

TABLE OF SPECIFICATIONS

Ordered from most to least stable:

	Cur- rent	Upcom- ing	Notes	
Completed				
CSS Snapshot 2018	NOTE		Latest stable CSS	
CSS Snapshot 2017	NOTE		Latest stable CSS	
CSS Snapshot 2015	NOTE			
CSS Snapshot 2010	NOTE			
CSS Snapshot 2007	NOTE			
CSS Color Level 3	REC	REC		
CSS Namespaces	REC	REC		
Selectors Level 3	REC	REC		
CSS Level 2 Revision 1	REC	REC	See Errata	
Media Queries	REC	REC		
CSS Style Attributes	REC	REC		
CSS Fonts Level 3	REC	REC		
CSS Basic User Interface Level 3	REC	REC		
Stable				
CSS Backgrounds and Borders Level 3	CR	PR		
CSS Conditional Rules Level 3	CR	CR		
CSS Multi-column Layout Level 1	WD	CR		
CSS Values and Units Level 3	CR	PR		
CSS Flexible Box Layout Level 1	CR	PR		
CSS Cascading and Inheritance Level 3	CR	PR		
CSS Writing Modes Level 3	CR	CR		
CSS Counter Styles Level 3	CR	PR		
Testing				
CSS Image Values and Replaced Content Level 3	CR	CR		
CSS Speech	CR	CR		
CSS Text Decoration Level 3	CR	CR		
CSS Shapes Level 1	CR	CR		
CSS Masking Level 1	CR	CR		
CSS Fragmentation Level 3	CR	PR		
CSS Cascading Variables	CR	PR		
Compositing and Blending Level 1	CR	CR		
CSS Syntax Level 3	CR	CR		
CSS Grid Layout Level 1	CR	PR		
CSS Display Level 3	CR	CR		

Afbeelding 8.1 Voltooide CSS-modules en een klein deel van de modules waaraan wordt gewerkt op www.w3.org/Style/CSS/current-work.

Waarom CSS zo handig is

Zelfs als u de dagen van HTML-opmaakelementen (``, `<center>`, `<blink>`) en HTML-opmaakattributen (`color`, `size`) niet hebt meegeemaakt, kunt u zich waarschijnlijk wel een voorstelling maken van de ellende. Alle opmaak moest in de HTML-code worden opgenomen en dat is een onderhoudsnachtmerrie. Het gaat immers niet om één pagina maar om complete websites. Lay-outs waren alleen mogelijk met tabellen genest tot vele niveaus diep en ook dat is nauwelijks onderhoudbaar.

De boodschap zal duidelijk zijn, CSS maakt het vormgeven en opmaken van webpagina's een stuk eenvoudiger (en leuker, veelzijdiger, ...):

- alle opmaak voor elementen met vergelijkbare kenmerken past in één stijlregel;
- de lay-out voor pagina's en componenten op die pagina's kan worden verdeeld over behapbare en logisch ingedeelde bestanden;
- de lay-out kan veel eenvoudiger worden aangepast door de ontwerper en kan zich automatisch aanpassen aan het weergaveapparaat;
- een CSS-bestand hoeft maar één keer te worden geladen en is dan beschikbaar voor alle HTML-pagina's;
- het HTML-bestand is schoner en beter leesbaar, omdat alle opmaakinformatie in het CSS-bestand staat.

Als er een nadeel van CSS zou moeten worden genoemd, is dat de groeiende complexiteit. Met CSS kan steeds meer, maar u moet ook steeds meer eigenschappen en waarden kennen. CSS-bestanden worden groter en moeilijker te lezen. Dat vraagt een gestructureerde manier van werken en goede documentatie. Dergelijke best practices komen aan het eind van dit hoofdstuk aan bod, maar het is goed als u zich hier van het begin af aan bewust van bent.

De taal CSS

Hoewel het werken met CSS lang niet altijd eenvoudig is, is de taal een toonbeeld van eenvoud. Elke stijl/regel bestaat uit drie dingen:

- selector het HTML-element dat wordt opgemaakt;
- eigenschap (property) het kenmerk dat wordt ingesteld (lettertype, grootte, kleur enzovoort);
- waarde (value) hoe het moet worden opgemaakt.



De taal CSS

CSS is een declaratieve taal, wat wil zeggen dat waarden worden toegewezen aan eigenschappen. Ter vergelijking: HTML is een markeringstaal en JavaScript is een programmeertaal.

	eigenschap	waarde	eigenschap	waarde
body	{	color: #FF99CC;	background:	#5E5E5E; }
selector	declaratie		declaratie	

Afbeelding 8.2 De structuur van een stijlregel met twee declaraties.

Een voorbeeld. Stel dat alle alineatekst moet worden weergegeven met het lettertype Roboto. We ontleden deze zin volgens CSS:

- *alineatekst* is de selector;
- *lettertype* is de eigenschap;
- *Roboto* is de waarde.

In de taal CSS wordt dit:

```
p {
    font-family: Roboto;
}
```

In deze code is:

- *p* de selector, in dit voorbeeld het HTML-element `<p>`;
- *font-family* de eigenschap;
- *Roboto* de waarde;
- de eigenschap en de waarde vormen samen de *declaratie*.

In dit voorbeeld bestaat de waarde uit tekst (de naam van het lettertype), maar het kan ook een getal (met een eenheid) zijn. Als diezelfde alineatekst een grootte van 20 pixels moet krijgen, wordt de stijlregel uitgebreid met een nieuwe declaratie:

```
p {
    font-family: Roboto;
    font-size: 20px;
}
```

In een stijlregel mag u zo veel declaraties opnemen als u wilt.

Behalve de selector, eigenschappen en waarden ziet u in de codevoorbeelden accolades, dubbelepunten en puntkomma's. Die zijn net zo belangrijk!

- Alle eigenschappen en waarden die bij een selector horen, komen tussen de accolades.
- Tussen de eigenschap en de waarde staat een dubbelepunt.
- Elk paar van een eigenschap en een waarde (een declaratie dus) wordt afgesloten met een puntkomma.



Witruimte wordt genegeerd

Spaties, tabs en regeleinden in CSS worden genegeerd, net zoals in HTML. U kunt daardoor een CSS-bestand net zo opmaken als u wilt, op een manier die u leesbaar vindt. De opmaak heeft geen invloed op de werking van de code.

Samengevat ziet een stijlregel er schematisch als volgt uit:

```
selector { eigenschap: waarde; }
```

- Selectors zijn er in allerlei variaties. Ze worden besproken in de paragraaf *Selectors*.
- CSS heeft ruim 350 eigenschappen en de belangrijkste daarvan worden in de komende hoofdstukken behandeld.
- Er zijn verschillende soorten waarden en daar gaat de volgende paragraaf over.

Waarden en eenheden

Elke eigenschap heeft mogelijke waarden, vaak meer dan één. De volgende waarden komt u regelmatig tegen:

- sleutelwoorden auto, content, flex-start, initial, block, left en dergelijke, gebruik ze zonder aanhalingstekens;

```
div {
  position: absolute;
}

p {
  text-align: right;
}
```

- getallen een getal of kommagetal, positief of negatief (het decimaalteken is in CSS net als in HTML de punt en de 0 wordt bij voorkeur weggelaten);

```
div {  
    opacity: .5;  
}
```

- lengte in CSS bestaat een lengte uit een getal met direct daarachter een eenheid zoals px of em – er mag geen spatie tussen staan;

```
div {  
    padding: 1em;  
}
```

- percentage een getal van 0 tot 100 met een procentteken (geen spatie ertussen);

```
img {  
    width: 100%;  
}
```

- URL een bestandslocatie, absoluut als url('http://www.domein.nl/images/plaatje.png') of relatief als url('../images/plaatje.png'); aanhalingstekens zijn niet verplicht;

```
div {  
    background-image: url('../images/plaatje.png');  
}
```

- kleur een kleurnaam of een numeriek waarde – er zijn diverse mogelijkheden voor het aangeven van kleuren en die worden in hoofdstuk 11 besproken.

```
div {  
    background-color: red;  
    background-color: rgb(255, 0, 0);  
    background-color: #ff0000,  
}
```

Globale waarden

Globale waarden is geen officiële term, maar er wordt mee bedoeld dat deze sleutelwoorden gelden voor elke eigenschap. Het zijn er drie:

- `initial` de beginwaarde (standaardwaarde) van een eigenschap volgens de specificatie.
- `inherit` hiermee krijgt (erft) de eigenschap de waarde van het ouderelement.
- `unset` de eigenschap erft de waarde (`inherit`) en als dat niet kan krijgt deze de beginwaarde (`initial`).

Lengte

Lengte is een belangrijk begrip in CSS. Bij een lengte gaat het om een afstand: een marge naar rechts, een rand omlaag, de breedte van een blok enzovoort. Ook de grootte van tekst is een lengte.

Een lengte bestaat uit een getal en een eenheid, behalve als de lengte 0 is; dan wordt eenheid weggelaten.

Een lengte-eenheid is absoluut of relatief. Bij het maken van webpagina's – die op een scherm in een browservenster worden bekeken, gebruikt u voornamelijk relatieve eenheden. De pixel is traditioneel de meestgebruikte absolute eenheid op beeldschermen, maar zelfs die wordt vaak vervangen door relatieve eenheden die rekening houden met schermgrootte en gebruikersvoorkieuren. Andere absolute eenheden zijn millimeter (mm) centimeter (cm), kwart millimeter (q), inch (in), pica (pc) en punt (pt). Deze kunnen nuttig zijn voor geprinte media (in een stijlblad voor het afdrukken van een pagina).



Alle eenheden

CSS heeft nogal wat toegestane eenheden. Een compleet overzicht is te vinden op www.w3.org/TR/css-values-3/.

De pixel

Oorspronkelijk was 1 CSS-pixel hetzelfde als 1 beeldschermpixel (hardware-pixel). Vanwege de variatie in schermen en resoluties definieert W3C nu de CSS-pixel als een 'optische referentie-eenheid'. Dat wil zeggen: een object dat van dichtbij wordt bekeken op een smartphone ziet er net zo groot uit als het van veraf wordt bekeken op een groot scherm. Wie vanaf vier meter naar een groot scherm kijkt en tegelijk op zijn smartphone op een normale afstand dezelfde webpagina bekijkt, moet tekst en vormen in min of meer dezelfde

grootte zien. De absolute afmeting van de pixel varieert dus, maar de kijkverring blijft hetzelfde. (Zie ook hoofdstuk 5 en hoofdstuk 10.)

In stylesheets wordt het woord pixel altijd afgekort tot px.

De eenheid em

Een tweede optie, voor tekst geschikter dan px, is de eenheid em. Em is geen afkorting, het is gewoon em. Het is ook een eenheid die soms tot verwarring leidt, als tekst in de browser een onverwacht formaat blijkt te krijgen.

Hoe groot is een em? Volgens de definitie is de em is gelijk aan de berekende tekstgrootte van het ouderelement. Dat lijkt nauwelijks een antwoord, maar de tekstgrootte van de ouder is altijd te achterhalen, als is het maar door die domweg op te zoeken met de Hulpmiddelen voor ontwikkelaars. Daarnaast hebt u één zekerheid: elke browser heeft dezelfde ingebouwde basisgrootte van tekst: 16 pixels. Met andere woorden: <html> en het kind <body> hebben als u zelf niets instelt een tekstgrootte van 16 pixels en alle kinderen van <body> overerfen diezelfde grootte. Stelt u deze grootte niet zelf in en baseert u alle verder afmetingen op de basisgrootte van het weergaveapparaat, dan geldt dus: 1em = 16px (en dat is ongeveer gelijk aan 14 punt). Alle andere tekstgrootten leidt u af van 1em, en een kop <h1> maakt u bijvoorbeeld 2.5em groot; dat zal standaard 40px zijn. Wilt u een grotere basisafmeting voor tekst, dan maakt u een declaratie voor het element <html> of <body> met een tekstgrootte van bijvoorbeeld 20 pixels (of 1.25em).

Als u weet wat de tekstgrootte van de ouder is, weet u hoe groot 1em van het kind is. Een kop <h1> van 2.5em en daarin een van 1em levert twee even grote teksten op. De <h1> is de ouder van de en daardoor erft de em van de zijn grootte van de <h1>. Voor het gewenste verschil in tekstgrootte (de half zo groot als de <h1>), moet de een tekstgrootte krijgen van 0.5em.

Meer over tekst en tekstgrootte in hoofdstuk 10.

De makkelijke em: rem

In de module *Values and Units Level 3* is een nieuwe eenheid geïntroduceerd: de rem. Deze eenheid refereert aan de tekstgrootte van het rootelement <html> (root em). Daarmee is de verwarring door overerving van em opgelost, want de grootte van rem verandert niet door de tekstgrootte van andere ouderelementen. Ook bij rem kunt u uitgaan van de tekstgrootte die het apparaat toewijst (16 pixels), maar u kunt zelf een tekstgrootte instellen op het element <html> en daarmee de gewenste leesbaarheid van de basistekst bereiken.

ken. Met de volgende regel wordt de standaardtekstgrootte ingesteld op 1.25rem, wat rem effectief de waarde $1.25 \times 16 = 20\text{px}$ geeft. Met andere woorden: tekst van 1rem wordt op deze manier 20px groot.

```
html { font-size: 1.25rem; }
```

Het handige van rem is dat u een tekstgrootte of andere afmeting altijd baseert op het rootelement en niet hoeft na te denken over de tekstgrootte van het ouderelement.



Discussie over px, em en rem

Er wordt veel gediscussieerd over het juiste gebruik van de eenheden px, em en rem. Niet alleen met betrekking tot tekstgrootte, maar ook als het gaat om breekpunten voor media queries (hoofdstuk 10) en de grootte van inhoudsblokken (hoofdstuk 9). Lees bijvoorbeeld *Use 'rem' for global sizing; use 'em' for local sizing*, inclusief de zijlijnen en het commentaar, op css-tricks.com/rem-global-em-local. Ook interessant is het artikel *PX, EM or REM Media Queries?* op zellwk.com/blog/media-query-units/.

Handig voor tekstvakken: ch

Waar em en rem draaien om de hoogte van tekst (de punigrootte), gebruikt de eenheid ch de breedte van tekst, om precies te zijn de breedte van het getal 0. Dat kan handig zijn voor het instellen van de breedte van bijvoorbeeld tekstkolommen of invoervakken voor tekst. Nu is een kenmerk van de meeste tekst dat de breedte van de tekens niet gelijk is. Als gevolg daarvan is bijvoorbeeld een invoervak voor een postcode (6 tekens en een spatie) met een breedte van 7ch voor normale tekst net iets te smal. Bij een monospace lettertype klopt het wel. Bij tekstkolommen is dit niet zo'n probleem; een teken meer of minder maakt dan niet zo veel uit.

Eenheden gebaseerd op de viewport

De module *Values and Units level 3* introduceerde ook eenheden gebaseerd op de viewport, zeg maar het browservenster. Met andere woorden: de indeling van de pagina, maar ook de grootte van tekst of andere inhoud, kan worden gekoppeld aan de grootte van het venster. Wordt het venster kleiner, dan schaalt de inhoud mee. Er wordt zelfs rekening gehouden met schuifbalken. Deze aanpassing is dynamisch: zodra de grootte van het venster verandert, veranderen de waarden.

Bij de grootte van het venster worden de schuifbalken niet meegerekend (behalve bij overflow: auto, zie hoofdstuk 9). De eenheden zijn:

- vw 1% van de breedte van het venster;
- vh 1% van de hoogte van het venster;
- vmin de kleinste waarde van vh en vw;
- vmax de grootste waarde van vh en vw.



Afbeelding 8.3 Formaat gebaseerd op venstergrootte: blokken met een hoogte van 33vh (een derde van de hoogte) en een breedte van 25vw, 50vw en 75vw.

Overige eenheden

Er zijn eenheden die u alleen in specifieke gevallen gebruikt, bijvoorbeeld bij radiale kleurverlopen (hoeken) of de duur van een animatie (tijd).

- hoeken: deg, grad, rad, turn
- tijdsduur: s, ms
- frequentie: Hz, kHz
- resolutie: dpi, dpcm, dppx

Samenvatting afmetingen

In de praktijk werkt u voor de afmetingen van blokken en de grootte van tekst met deze eenheden en waarden:

- percentage
- px

- em
- rem
- vw, vh, vmin, vmax



Kijk naar de toekomst

De volgende versie van de CSS-module voor waarden en eenheden (*CSS Values and Units Module level 4*) is nog maar een concept, maar geeft wel een beeld van wat de toekomst zou kunnen brengen: www.w3.org/TR/css-values-4/.

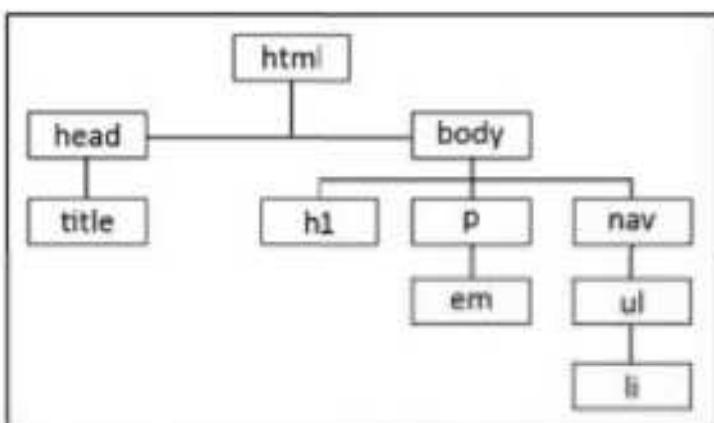
CSS-verwerking door de browser

De boomstructuur

Nadat de browser een HTML-document heeft geladen, wordt een boomstructuur (document tree) van alle elementen gemaakt. Die boomstructuur weerspiegelt de relaties tussen alle HTML-elementen. Elk element heeft één ouder-element – behalve het rootelement <html>. Kernbegrippen zijn:

- ouder (*parent*) direct bovenliggend element;
- kind (*child*) direct onderliggend element;
- broer/zus (*sibling*) element op hetzelfde niveau;
- voorouder (*ancestor*) bovenliggend element, maar niet direct;
- afstammeling (*descendant*) onderliggend element, maar niet direct.

De Engelse benamingen zijn nuttige kennis voor het begrijpen van de meestal Engelstalige documentatie op internet. De relaties komt u ook tegen in de paragraaf *Selectors* (selectors gebaseerd op afstamming).



Afbeelding 8.4 De boomstructuur van het HTML-document geeft een beeld van de relaties tussen de elementen. U kunt de boomstructuur van elke webpagina bekijken met de Hulpmiddelen voor ontwikkelaars.

Bekijk het codevoorbeeld en afbeelding 8.4. Let op het niveau waarop de elementen staan.

```
<html>
  <head>
    <title>De HTML-boomstructuur</title>
  </head>
  <body>
    <h1>Een kop</h1>
    <p>Een stuk tekst met <em>nadruk</em> op een woord.</p>
    <nav>
      <ul>
        <li>Home</li>
        <li>Sectie 1</li>
      </ul>
    </nav>
  </body>
</html>
```

- het rootelement `<html>` staat boven in de hiërarchie en heeft geen ouder. Het is wel de ouder van `<head>` en `<body>`;
- `<body>` is de ouder van `<h1>`, `<p>` en `<nav>`;
- `<p>` is de ouder van ``;
- `<nav>` is de ouder van ``;
- `` is de ouder van elke ``;
- ``-elementen in dezelfde lijst zijn siblings (kinderen van dezelfde ouder, er bestaat geen Nederlands synoniem);
- ook `<h1>`, `<p>` en `<nav>` zijn siblings, met als onderscheid dat `<h1>` en `<p>`, en `<p>` en `<nav>` aangrenzende elementen zijn (*adjacent elements*); `<h1>` en `<nav>` zijn wel siblings, maar niet aangrenzend.

Waarden toewijzen

Wanneer de boomstructuur is opgebouwd, krijgen alle eigenschappen van alle elementen waarden toegewezen. Daarbij gaat de browser als volgt te werk:

- Eerst wordt gekeken of een waarde voor de eigenschap in de cascade (zie hierna) beschikbaar is, dat wil zeggen:
 - een stijlblad van de auteur (de webontwikkelaar);
 - een stijlblad van de gebruiker;
 - een stijlblad van de browser.
- Zo niet, dan wordt gekeken of de opmaakinformatie kan worden overgenomen van het ouderelement. (Dit heet overerving of *inheritance* en wordt hierna uitgelegd.)

- Kan dat ook niet, dan wordt de beginwaarde (initiële waarde) van de eigenschap gebruikt.



Stijlblad van de gebruiker

Iemand die uw HTML-pagina bekijkt, heeft (in theorie) de mogelijkheid om uw opmaak te overschrijven door zijn eigen stijlblad te laden met bijvoorbeeld instellingen voor grote tekst of hoog contrast. Internet Explorer had deze optie, maar in moderne browsers ontbreekt die veelal. Wel kunnen in elke browser het lettertype, de teken-grootte en de kleuren worden aangepast. Deze instelling kan die van de website overschrijven.

De cascade

Opmaakinstructies worden in een vaste volgorde opgezocht en verwerkt. Die volgorde is waar CSS zijn naam aan te danken heeft: de cascade. Nu weet u uit eigen ervaring en na de toelichting in de vorige paragraaf dat elk HTML-element sowieso een bepaald uiterlijk heeft, vastgelegd in het stijlblad van de browser. Dat uiterlijk komt tot stand tijdens het zoeken naar opmaakkenmerken in diverse bronnen, totdat de laatste en belangrijkste bron is gevonden. De stappen van de browser zijn:

- 1 Zoek voor een element alle geldige declaraties op. Geldig wil zeggen dat de selector overeenkomt met het element en dat het weergavemedium overeenkomt met de eigenschap media.
- 2 De bronnen voor opmaakkenmerken zijn in oplopende volgorde van belangrijkheid:
 - declaraties in de browser;
 - declaraties van de gebruiker;
 - declaraties van de auteur;
 - declaraties van de auteur met kenmerk !important;
 - declaraties van de gebruiker met kenmerk !important.



Pas op met !important

Het kenmerk !important is een wolf in schaapskleren. Met dit kenmerk wordt een stijlregel belangrijker dan elke andere declaratie voor dat element. Dat lijkt een makkelijke oplossing voor een opmaakprobleem, maar pas op. Is een eigenschap eenmaal !important, dan kunt u er alleen nog overheen met weer een !important. Het gevaar dat u verzandt in een stijlblad vol belangrijke regels is groot; het oorspronkelijke probleem oplossen heeft verreweg de voorkeur. Het begrijpen van specificiteit (zie hierna) helpt daarbij enorm.

- 3 Sorteer de declaraties uit dezelfde bron en met dezelfde belangrijkheid op specificiteit. Dit onuitspreekbare begrip betekent: hoe specifiek/nauwkeurig is de selector die is gebruikt om het element te benaderen. De hoogste specificiteit wint. Het begrip wordt hierna verder uitgelegd.
- 4 Sorteer op de volgorde waarin de declaratie is aangegeven. Als twee declaraties gelijkwaardig zijn en uit dezelfde bron komen, wint de laatste. Met ander woorden: als u in een stijlblad een eigenschap twee keer een waarde geeft, wordt de laatst toegewezen waarde gebruikt.

Specificiteit berekenen

Specificiteit is een numerieke weergave van hoe specifiek een selector is. Daarmee kunt u achterhalen welke stijldeclaratie wordt toegepast wanneer opmaakregels met elkaar concurreren. Let op, opmaakregels kunnen pas concurrenten zijn wanneer de declaraties uit dezelfde bron komen en even belangrijk zijn (al dan niet beide !important).

Op het moment dat u een CSS-eigenschap instelt, het maar niet lukt om de wijziging doorgevoerd te krijgen en u geen typefout hebt gemaakt, is het tijd om de specificiteit na te gaan. Start de Hulpmiddelen voor ontwikkelaars, selecteer het onwillige element en bekijk de bijbehorende CSS-stijlregels. Meestal is specificiteit het probleem.

Specificiteit wordt uitgedrukt in een reeks van vier getallen. De specificiteit neemt af van links naar rechts. De berekening wordt als volgt gemaakt:

- Elk id-attribuut telt voor 0,1,0,0.
- Elk class-attribuut, elk ander attribuut en elke pseudoklasse telt voor 0,0,1,0.
- Elke elementnaam en elk pseudo-element telt voor 0,0,0,1.
- De universele selector * telt niet mee.

In dit overzicht ontbreekt 1,0,0,0. Die waarde krijgt u alleen als de declaratie in het attribuut style van het element staat, dus in het HTML-document. Op dat moment hoeft u niet verder te kijken, want deze declaratie gaat boven alle andere.

Uit het resultaat is eenvoudig af te leiden welke declaratie voorrang heeft: bekijk van links naar rechts of u een waarde tegenkomt. Dus tel bij de selector eerst het aantal id-attributen, dan het aantal class-attributen en daarna alle elementselectors en pseudo-elementen. Zodra een kolom niet 0 is, kijkt u of er in diezelfde kolom een hogere waarde staat. De declaratie met de hoogste waarde in die kolom wordt toegepast. Het doet er niet toe hoe hoog de waarden van de kolommen erna zijn; die tellen niet meer mee. Extreem voorbeeld:

als een selector bestaat uit elf elementen maar een concurrerende declaratie heeft één klasseselector, dan telt de klasseselector.

Zijn er in een kolom wel gelijke waarden, een element heeft bijvoorbeeld twee stijlregels waarin beide een ID is gebruikt, dan pas kijkt u naar de volgende kolom, en zo verder.

Bekijk de voorbeelden in de tabel. Elk vetgedrukte selector heeft een hogere specificiteit dan die ervoor.

Selector	Specificiteit	Toelichting
div ul ul li	0,0,0,4	4 elementsselectors
div.klasse ul li	0,0,1,3	1 klasseselector, 3 elementsselectors
a::before	0,0,0,2	1 pseudoklasse, 1 elementselector
a.klasse:hover	0,0,2,1	1 klasse, 1 pseudoklasse, 1 elementselector
p.eerste	0,0,1,1	1 klasseselector, 1 elementselector
#id p.eerste	0,1,1,1	1 id-selector, 1 klasseselector, 1 elementselector

- Opdracht Bekijk het volgende voorbeeld en beantwoord de vraag: wordt `` cursief of normaal?

```
/* CSS-declaraties */
p em {
    font-style: italic;
}

article em {
    font-style: normal;
}

<!-- HTML-code -->
<html>
<head>
    <title></title>
</head>
<body>
    <article>
        <h2>Paragraaf 1</h2>
        <p>Ik heb nog zo gezegd: <em>geen hammetje!</em></p>
    </article>
</body>
</html>
```

Het antwoord is: normaal. Beide declaraties hebben twee elementselectors en specificiteit 0,0,0,2. Omdat ze gelijkwaardig zijn, wint de laatstgenoemde declaratie. Dat de eerste 'dichterbij' staat, is onbelangrijk. Bij specificiteit gaat het alleen om de numerieke waarde en als die gelijk is, wint de laatstgenoemde.

Nogmaals, herlees deze paragraaf nadat u de informatie over selectors hebt doorgenomen. Het begrip specificiteit zal daardoor meer gaan leven.

Overerving

Een opmaakeigenschap hoeft niet rechtstreeks een waarde te krijgen. De waarde kan worden overgeerfd van het ouderelement (*inheritance*). De gedachte er achter is dat op deze manier niet elke eigenschap voor elk element afzonderlijk hoeft te worden ingesteld als toch al is te verwachten dat die hetzelfde zal zijn als van zijn ouder. Daardoor kunnen stijlbladen kleiner blijven. Het is wel belangrijk om het principe van overerving te begrijpen, omdat u anders voor verrassingen kunt komen te staan bij het vormgeven van elementen.

Overerving is gebaseerd op het overnemen van de berekende waarde van het ouderelement. Dat is niet altijd de waarde die u in het stijlblad ziet (zelf hebt ingesteld), want ook het ouderelement kan een berekende waarde hebben die bestaat uit overerving en ingestelde waarde. Dit doet zich vooral voor bij eenheden waarvan het resultaat doorwerkt, zoals procenten of em. Hoewel de details daarvan later nog aan de orde komen, kunt u zich zeker een voorstelling maken bij de volgende CSS- en HTML-code:

```
/* de CSS-declaraties */
body {
    font-size: 20px;
}
section {
    font-size: 1.5em;
}
p {
    font-size: 1em;
}
<!-- de HTML-code -->
<body>
    <p>Deze tekst is 1em = 20px van de ouder: body.</p>
    <section>
        <p>Deze alinea heeft ook een tekstgrootte van 1 em, maar dat is nu 24px. Reden: de ouder is section en die heeft een tekstgrootte van 1.5em = 30px (1,5 maal die van de ouder van section, body).</p>
    </section>
</body>
```

Deze tekst is 1em = 20px van de ouder: body.

Deze alinea heeft ook een tekstgrootte van 1 em, maar dat is nu 30px. Reden: de ouder is section en die heeft een tekstgrootte van 1.5em = 30px (1,5 maal die van de ouder van section, body).

Afbeelding 8.5 Overerving en het effect op relatieve eenheden (08_05.html).

Niet elke eigenschap wordt automatisch overgeërfd, maar elke eigenschap kan de waarde `inherit` krijgen. Daarmee geeft u aan dat de waarde hoe dan ook moet worden overgenomen van de ouder. U kunt dus ook eigenschappen door overerving doorgeven als dat standaard niet gebeurt.

Voor CSS 2.1 bestaat een tabel met de eigenschappen die standaard via overerving worden doorgegeven aan kindelementen (www.w3.org/TR/CSS21/propidx.html), maar langzamerhand worden de CSS 2-eigenschappen opgenomen in CSS3-modules. Van de CSS3-modules bestaat niet een compleet overzicht met alle overervende eigenschappen. Wel bevat elke module een overzicht van de bijbehorende eigenschappen (*property index*) met daarbij onder meer of de eigenschap overerft. Gezond verstand helpt ook; het is niet erg logisch om een rand of een achtergrondafbeelding standaard te laten overerven. De aan tekst gerelateerde eigenschappen overerven allemaal: `font-`, `letter-`, `line-height`, `list-`, `orphans`, `quotes`, `tab-size`, `text-`, `white-space`, `widows`, `word-`.

Percentages werken door

Ook bij percentages bepaalt de ouder het effect van een waarde bij het kind-element. In het volgende voorbeeld zijn er drie geneste blokken, waarvan de binnennste twee een hoogte en breedte van 50% hebben. Het tweede blok is daardoor half zo hoog en half zo breed als het buitenste blok. Het derde blok is de helft van zijn ouder (het tweede blok) en dat is maar een zesde van het buitenste blok. Zie afbeelding 8.6.

```
div.inner {
    height: 50%;
    width: 50%;
}
<div>
    <div class="inner">
        <div class="inner"></div>
    </div>
</div>
```



Afbeelding 8.6 Percentages zijn gerelateerd aan de ouder (het 'containing block') (08_06.html).

De plaats van de CSS-declaraties

De stijldeclaraties voor een HTML-document kunnen op drie plaatsen staan:

- in het attribuut `style` van het element;
- in het element `<style>` in de `<head>` van het HTML-document;
- in een afzonderlijk CSS-bestand.

In het algemeen heeft een afzonderlijk CSS-bestand de voorkeur. Er kunnen redenen zijn om een van de andere mogelijkheden te kiezen, bijvoorbeeld voor het testen van opmaak of voor heel specifieke instellingen.

Denk nog even terug aan de specificiteit: het attribuut `style` bij een element gaat boven alles; bij het element `<style>` en een externe stylesheet gaat het om specificiteit – als die berekening geen winnaar oplevert, telt de laatstgenoemde declaratie.



Deel losse CSS toch weer bij HTML

Door het streven naar korte laadtijden van webpagina's wordt steeds vaker een deel van de CSS in het HTML-bestand geplaatst. Daarmee wordt voorkomen dat het laden van een (groot) afzonderlijk CSS-bestand het opbouwen en weergeven van de HTML-pagina vertraagt. De meeste CSS wordt nog steeds in een of meer afzonderlijke style-sheets geschreven, maar de kritische CSS, nodig voor de weergave van het eerst zichtbaar deel van de pagina (het deel 'boven de vouw'), wordt in de eindfase van de productie aan het HTML-bestand toegevoegd.

Het attribuut style

Een van de globale HTML-attributen is `style`. Het biedt de mogelijkheid om rechtstreeks bij het element CSS-opmaakmerken toe te wijzen:

```
<p style="font-family: Lato; font-size: 20px;"> ... </p>
```

Deze stijl wordt een regelstijl (*inline style*) genoemd en is het belangrijkst (hoogste specificiteit). In de cascade gaat deze boven opmaak die in het element `<style>` of in het CSS-bestand staat. Het probleem is dat dergelijke stijlen lastig te onderhouden zijn. De opmaak komt verspreid in het HTML-document te staan, wat het onderhoud bijzonder lastig maakt.

Advies: niet gebruiken of hooguit om instellingen te testen.

Het element `<style>`

Met het element `<style>` kan CSS-opmaak in de `<head>` of de `<body>` van het HTML-document worden opgenomen. Dit worden ingesloten stijlen genoemd (*embedded styles*). Het element heeft drie niet-verplichte attributen die meestal ook niet worden gebruikt, en uiteraard de globale attributen:

- `media` voor welke weergave de stijlregels gelden, bijvoorbeeld beeldscherm, printer, een bepaalde schermgrootte enzovoort (zie ook hoofdstuk 12, de paragraaf *Media queries*). De beginwaarde (die u mag weglaten) is `all` en dat betekent dat de stijlregels voor alle media gelden.
- `type` specificeert de opmaaktaal. De standaardwaarde is `text/css` en daarom kunt u het attribuut weglaten.
- `title` het globale attribuut `title` benoemt in dit geval alternatieve stijlbladen. Daarmee kan een gebruiker dezelfde pagina bekijken met een andere opmaak, bijvoorbeeld een weergave met sterk contrasterende kleuren.

```
<style>
p {
  font-family: Lato;
  font-size: 20px;
}
</style>
```

In de cascade komt deze methode op de tweede plaats. De opmaakininstellingen zijn één stap minder belangrijk dan die in het attribuut `style`.



Mag in <body>, <head> is beter

Dat het element `<style>` ook in de `<body>` mag worden geplaatst, is nieuw in HTML 5.2. In de specificatie wordt wel gezegd dat `<style>` in de `<head>` nog steeds de voorkeur heeft. De browsers verwerken `<style>` in de `<body>` net zo goed als `<style>` in de `<head>`.



Stijlblad importeren

Voor de volledigheid: een stijlblad kan ook worden geïmporteerd met de instructie `@import url('css/stijlen.css')` in een `<style>`-element. Er is geen goede reden om dit te doen, maar het kan. U kunt net zo goed `<link>` gebruiken.

Advies: gebruik het element `<style>` bij voorkeur alleen voor opmaak van inhoud boven de vouw. Het kan ook worden gebruikt als een pagina een aantal heel specifieke opmaakmerken heeft, afwijkend van de opmaak van de andere pagina's in de site. In dat geval kan importeren met `@import` handig zijn.

Extern bestand: `<link>`

De derde optie is het verzamelen van alle opmaakmerken in een afzonderlijk bestand (eventueel meerdere bestanden). Deze methode heeft de voorkeur. Voor het koppelen van een extern CSS-bestand gebruikt u het element `<link>`. Ook dit element moet u opnemen in de `<head>` van het document. De volledige uitleg vindt u in hoofdstuk 4. Samengevat: het attribuut `rel` geeft aan dat een stylesheet wordt gekoppeld en `href` bevat de URL van het stijlenbestand:

```
<head>
  <title>CSS - basis</title>
  <link rel="stylesheet" href="css/stijlen.css">
</head>
```

Met het attribuut `media` kan worden aangegeven voor welk apparaat en voor welke schermgrootte een stylesheet is bestemd. Het is ook mogelijk om in de stylesheet zelf met media queries aan te geven onder welke omstandigheden een bepaalde opmaak geldt (zie hoofdstuk 13, de paragraaf *Media queries*).



Alles wordt geladen

Ook wanneer u met het attribuut `media` aangeeft dat een stijlblad alleen geldt voor bijvoorbeeld afdrukken (`media="print"`), wordt toch het stijlblad geladen. De browser kan niet vooraf bepalen of een stijlblad al dan niet nodig is en laadt daarom alle opmaakinfo die u beschikbaar stelt.

Het CSS-bestand

Een CSS-bestand is een tekstdocument met platte tekst (dus geen docx- of rtf-bestand). De extensie moet `.css` zijn. Voor editors geldt hetzelfde als bij HTML: elke editor voor platte tekst voldoet en gebruik geen tekstverwerkers zoals Word. Elke moderne HTML-editor heeft ook opties om goed met CSS te kunnen werken. De taal wordt herkend en vaak zijn er opties voor automatisch aanvullen van code, overzichtelijk inspringen, het plaatsen van accolades en kleurcodering. In hoofdstuk 1 vindt u meer informatie over editors.

Commentaar toevoegen

Commentaar is een niet te onderschatten onderdeel van uw CSS-bestand. Commentaar wordt gemarkerd met `/*` en `*/`. Alles wat daartussen staat, wordt door de browser genegeerd. Commentaar mag meerdere regels lang zijn en kan elk teken bevatten. Gebruik het om het bestand documenteren. Wat op het moment waarop u het schrijft logisch lijkt, kan later een puzzel blijken en dat voorkomt u door te noteren wat de code doet.

```
/* Deze tekst wordt door de browser genegeerd */
*****
Deze tekst wordt ook genegeerd
*****
```

Commentaar kan ook worden gebruikt om (tijdelijk) complete stijlregels of losse declaraties uit te schakelen:

```
/* De hele stijlregel voor p is uitgeschakeld
p {
    font-family: Lato;
    font-size: 16px;
}
*/
p {
```

```
font-family: Lato;  
/* font-size: 16px; alleen deze regel is uitgeschakeld */  
|
```

Indeling van CSS-bestanden

Naarmate CSS-bestanden groter worden, wordt een overzichtelijke indeling belangrijker. Dat kan op allerlei manieren. Denk aan blokken met globale opmaakregels, regels voor tekst of definities van variabelen. Verder kan opmaak per paginaonderdeel of component worden gebundeld. Plaats bijvoorbeeld alle opmaak voor de header, de footer en de navigatie bij elkaar. Dit kan ook worden gedaan met de opmaakregels voor formulieren, blokken met gelijkvormige inhoud (kaarten of cards), de opmaak van een zijbalk enzovoort. Met commentaar kunnen de blokken worden gedocumenteerd en van elkaar worden gescheiden. Voor de indeling bestaan geen regels, maar er wordt door webbouwers wel veel over nagedacht en geschreven. Kijk eens rond hoe anderen het doen, lees artikelen en probeer indelingen uit om een methode te vinden die u bevalt. Leestip: www.danylkoweb.com/Blog/5-methodologies-for-architecting-css-ip.

Andere stijlbladen importen

De regel @import is hiervoor al genoemd, als onderdeel van het element <style>. Met @import kan ook een stylesheet in een externe stylesheet worden geimporteerd. Ook nu geldt dat net zo goed <link> in het HTML-document kan worden gebruikt. De regel @import url('bestand.css') moet de eerste instructie in de stylesheet zijn, dus voor alle andere stijlregels staan.

De ingebouwde stijlen aanpassen

Elke browser bevat een ingebouwd stijlblad om de HTML enige opmaak te geven. Dat is vooral handig als om wat voor reden dan ook de stylesheets niet worden geladen. Dankzij de ingebouwde stijlen krijgen bezoekers niet een pagina te zien met alleen maar platte tekst, maar zien zij een (eenvoudige maar duidelijke) pagina met koppen, lijsten enzovoort.

Het vervelende is dat niet elke browser dezelfde basisinstellingen heeft. Eigenschappen die niet in een stijlblad van de webontwikkelaar zijn ingesteld, kunnen daardoor per browser verschillen. Omdat elke ontwikkelaar dit probleem heeft, zijn er allerlei CSS-bestanden ontwikkeld die ervoor zorgen dat elke browser begint met gestandaardiseerde instellingen. U vindt ze als *CSS reset* of *CSS normalizer*. Er is een verschil tussen die twee, want een reset kan zo ver gaan als het strippen van alle opmaak, terwijl 'normaliseren' een wat fijnzinni-

```
font-family: Lato;  
/* font-size: 16px; alleen deze regel is uitgeschakeld */  
]
```

Indeling van CSS-bestanden

Naarmate CSS-bestanden groter worden, wordt een overzichtelijke indeling belangrijker. Dat kan op allerlei manieren. Denk aan blokken met globale opmaakregels, regels voor tekst of definities van variabelen. Verder kan opmaak per paginaonderdeel of component worden gebundeld. Plaats bijvoorbeeld alle opmaak voor de header, de footer en de navigatie bij elkaar. Dit kan ook worden gedaan met de opmaakregels voor formulieren, blokken met gelijkvormige inhoud (kaarten of *cards*), de opmaak van een zijbalk enzovoort. Met commentaar kunnen de blokken worden gedocumenteerd en van elkaar worden gescheiden. Voor de indeling bestaan geen regels, maar er wordt door webbouwers wel veel over nagedacht en geschreven. Kijk eens rond hoe anderen het doen, lees artikelen en probeer indelingen uit om een methode te vinden die u bevalt. Leestip: www.danylkoweb.com/Blog/5-methodologies-for-architecting-css-ip.

Andere stijlbladen importen

De regel @import is hiervoor al genoemd, als onderdeel van het element <style>. Met @import kan ook een stylesheet in een externe stylesheet worden geimporteerd. Ook nu geldt dat net zo goed <link> in het HTML-document kan worden gebruikt. De regel @import url('bestand.css') moet de eerste instructie in de stylesheet zijn, dus voor alle andere stijlregels staan.

De ingebouwde stijlen aanpassen

Elke browser bevat een ingebouwd stijlblad om de HTML enige opmaak te geven. Dat is vooral handig als om wat voor reden dan ook de stylesheets niet worden geladen. Dankzij de ingebouwde stijlen krijgen bezoekers niet een pagina te zien met alleen maar platte tekst, maar zien zij een (eenvoudige maar duidelijke) pagina met koppen, lijsten enzovoort.

Het vervelende is dat niet elke browser dezelfde basisinstellingen heeft. Eigenschappen die niet in een stijlblad van de webontwikkelaar zijn ingesteld, kunnen daardoor per browser verschillen. Omdat elke ontwikkelaar dit probleem heeft, zijn er allerlei CSS-bestanden ontwikkeld die ervoor zorgen dat elke browser begint met gestandaardiseerde instellingen. U vindt ze als *CSS reset* of *CSS normalizer*. Er is een verschil tussen die twee, want een reset kan zo ver gaan als het strippe van alle opmaak, terwijl 'normaliseren' een wat fijnzinni-

ger oplossing is die alleen de echte problemen aanpakt. Meestal is een reset ook niet bedoeld om zonder aanpassingen te gebruiken. Door het bestand aan te passen aan het beoogd gebruik voorkomt u dat de opmaak van elk element van uw eigen stijlinstellingen afhankelijk wordt.

We noemen drie bekende oplossingen, twee resets en een normalisatiebestand. Lees de toelichting bij de bestanden voordat u ermee gaat werken.

De aanbevolen keus is normalize.css. Dit bestand wordt regelmatig bijgewerkt en is daardoor geschikter voor modern webontwerp (en het is minder ingrijpend).

- **Reset CSS** De oer-reset van CSS-specialist Eric Meyer: meyerweb.com/eric/tools/css/reset/.
- **HTML5 reset stylesheet** Een door HTML5 Doctor aangepaste versie van Reset CSS: html5doctor.com/html-5-reset-stylesheet/.
- **normalize.css** Het bekendste normalisatiebestand voor CSS: nicoalagher.com/about-normalize-css/.

Om een van deze bestanden te kunnen gebruiken, moet u het downloaden en in de map met CSS-bestanden opslaan (en uploaden naar de webserver) als CSS-bestand. Voeg in de `<head>` van het HTML-document een `<link>` toe voor een verwijzing naar het bestand, bijvoorbeeld:

```
<head>
<title>CSS - basis</title>
<link rel="stylesheet" href="css/normalize.css">
<link rel="stylesheet" href="css/uw-eigen-stijlblad.css">
</head>
```

Het is belangrijk dat uw eigen stijlblad wordt geladen na normalize.css. De CSS-regels worden uitgevoerd in de volgorde waarin ze worden geladen en zo overschrijft u niet uw eigen regels.



Analyseer normalize.css

Wanneer u wat meer thuis bent in CSS, is het erg leerzaam om het bestand normalize.css grondig door te nemen. U leert de eigenaardigheden van de diverse browsers kennen en krijgt meer inzicht in de opmaak van 'bijzondere' elementen zoals `<input>` of `<button>` (die soms verrassend lastig zijn aan te passen).

Elementen benaderen met selectors

In hoofdstuk 2 wordt uitgelegd dat van het HTML-document een boomstructuur wordt gemaakt, het *document object model* (DOM). Het is uw taak om in het DOM het juiste element te selecteren om dat te kunnen opmaken met CSS.

Daarvoor dienen selectors. Een selector selecteert op basis van uw aanwijzingen een element en past daar de CSS-opmaakeigenschappen uit de stijlregel op toe.

Zoals met zo veel onderdelen van CSS zijn de selectors volop in ontwikkeling. *Selectors level 3* is de standaard (W3C-aanbeveling sinds november 2018) en wat daarin staat wordt hier besproken. *Selectors level 4* is nog een concept. Er staan een aantal nieuwe selectors in, maar voor praktische toepassing is het (eerste helft 2019) nog te vroeg.

Selectors zijn ze globaal verdeeld in de volgende groepen:

- basise Selectors;
- combinatie Selectors;
- pseudoklassen;
- pseudo-elementen.

Met selectors kunt u bijvoorbeeld een element selecteren dat afstamt van een ander element, maar er niet per se een kind van is. Of u geeft aan dat het juist wel een kind moeten zijn of dat het per se direct na een bepaald element moet komen. De mogelijkheden gaan echter veel verder. U kunt selecteren op attributen (bijvoorbeeld alle invoervelden van het type checkbox), op een toestand (een element heeft de focus), op de plaats van een element in de code (het eerste, vijfde of laatste item in een lijst) enzovoort.

Er is echter één beperking. U kunt in de boomstructuur alleen omlaag selecteren. Het is bijvoorbeeld niet mogelijk om van een element het ouderelement te selecteren. Bekijk de volgende code:

`<p>...</p>` kunt in de boomstructuur alleen omlaag selecteren.

- U kunt niet zeggen: selecteer de alinea <p> als daarin een element staat.
- U kunt wel zeggen: selecteer het element dat in een alinea <p> staat. (De code hiervoor is `p > em` of `p em`).

Eigenlijk is omlaag selecteren niet wat er gebeurt. De browsers begint namelijk aan de rechterkant van de selectors. In de selector `p > em` zoekt de browser

elke `` in het document op en kijkt dan of die in een `<p>` staat. Als dat zo is, wordt de opmaak toegepast.

Elementnaam, klasse of beide?

Voordat wordt uitgelegd welke selectors er zijn en hoe ze worden gebruikt, is het goed om eerst te kijken naar de beschikbare mogelijkheden:

Elementnamen en selectors

In een gestructureerd en goed gemarkerd HTML-document kunt u een groot deel van de pagina opmaken met alleen de basisselectors. Headers en footers, secties en artikelen, main en aside, het zijn stuk voor stuk ijzersterke haakjes voor de CSS. Verfijning brengt u aan met combinatieselectors, pseudoklassen en pseudo-elementen. Zo kan bijvoorbeeld een menu worden opgemaakt met alleen elementnamen en een algemene afstammingsselector (de spatie):

```
<nav>
  <ul>
    <li>
      <a>...</a>
    </li>
    <li>
      <a>...</a>
    </li>
  </ul>
</nav>
/* CSS */
nav { ... }
nav ul { ... }
nav li { ... }
nav a { ... }
```

Zonder heel specifiek te worden, wordt de opmaak van de lijst `ul`, de lijstitems `li` en de links `a` beperkt tot alleen elementen die afstammen van een `nav`. Het is niet nodig om de hele boomstructuur te doorlopen om bijvoorbeeld de link in een menu te bereiken. De volgende regel is veel te uitgebreid (`is` de kindselector):

```
nav > ul > li > a { ... } /* onnodig specifiek */
```

Zijn er twee menu's op de pagina, dan is er ook nog een selector om te bepalen of de opmaak geldt voor de eerste of tweede nav. Als het menu in het voorbeeld de hoofdnavigatie is en als eerste in de HTML-code staat, wordt de CSS:

```
nav:first-of-type { ... }
nav:first-of-type ul { ... }
nav:first-of-type li { ... }
nav:first-of-type a { ... }
```

Een tweede menu wordt dan aangeduid met `nav:nth-of-type(2)` en als het 't laatste menu is kan het ook met `nav:last-of-type`.

Op een vergelijkbare manier kunnen alinea's in een artikel een andere opmaak hebben dan die in een footer of sidebar. De algemene afstammingssselector (de spatie) doet het werk:

```
article p { ... }
```

Dit zijn eenvoudige voorbeelden, maar ook als het ingewikkelder wordt kan een HTML-pagina worden opgemaakt met elementnamen en de uitgebreide sortering geavanceerdere selectors.

Er is ook een keerzijde. De CSS is op deze manier volledig gekoppeld aan de HTML-structuur. Als een (onderdeel van) de pagina moet worden aangepast en daardoor de structuur van de HTML verandert, zullen de CSS-selectors niet meer kloppen (hoewel in de praktijk na een aanpassing van de HTML vaak ook de CSS moet worden aangepast). Ook kan het complexer worden. Als bijvoorbeeld het menu moet worden uitgebreid met een submenu, wordt de HTML-code:

```
<nav>
  <ul>
    <li>
      <a>...</a>
    <ul>
      <li>
        <a>...</a>
      </li>
    </ul>
  </li>
  <li>
    <a>...</a>
  </li>
</ul>
```

```
</nav>

/* CSS */
nav { ... }
nav ul { ... }
nav li { ... }
nav a { ... }
nav li > ul { ... } /* submenu */
nav li li { ... }
nav li li > a { ... }
```

Ondanks de diepere nesting kunnen alle elementen worden geselecteerd. Daarbij zullen er in werkelijkheid meer haakjes voor CSS zijn, bijvoorbeeld attributen en specifieke combinaties van elementen. Deze manier om de CSS aan de HTML te koppelen heeft fanaticieke voorstanders, maar ook net zulke fanaticie tegenstanders. Hoe doen de tegenstanders het dan?

Klassen

Toen HTML-pagina's voornamelijk uit betekenisloze div's werden opgebouwd, was het noodzakelijk om met klassen (en ID's) de div's van elkaar te onderscheiden. Daar kregen de div's niet meer betekenis van, maar de ontwikkelaar (en anderen die aan de code moesten werken) kon in elk geval zien om welke onderdeel het ging. Die noodzaak is nu een stuk minder, maar toch is de klasse een veelgebruikte selector, ook voor duidelijke HTML-elementen zoals `<header>`, `<nav>` en `<footer>`. Dat komt mede doordat veel sites en apps worden gebouwd met CSS-bibliotheken zoals Bootstrap, Material Design Components, Buima, Pure CSS en veel, heel veel andere. Ook JavaScript-frameworks (met jQuery voorop) hebben hier grote invloed op gehad – en nog steeds. Zo'n bibliotheek bestaat uit CSS-klassen met uitgewerkte opmaak, die eenvoudig in een HTML-document kunnen worden toegepast... met een klassenaam. (Zo'n bibliotheek bevat vaak ook JavaScript om de componenten functionaliteit te geven.) Ook zijn er CSS-naamgevingsmethoden die het gebruik van klassen doorvoeren tot in de kleinste details (Atomic CSS) of een iets minder gedetailleerd niveau (BEM, OOCSS, SMACCS).

En – laten we eerlijk zijn – overal een klasse aan hangen is ook wel lekker makkelijk, zelfs als er geen CSS-bibliotheek wordt gebruikt. Waarom leren werken met al die lastige selectoren? Het gevaar van deze benadering is dat de specificiteit te groot wordt. Als voor elke stukje opmaak een klasse wordt gemaakt, wordt het wel heel lastig om daar nog overheen te gaan zonder !important in te zetten. Het gebruik van klassen vraagt daarom net zo veel zorgvuldigheid als het gebruik van elementselectors.

Klassen hebben ook serieuze voordelen. Een voorbeeld is het beperken van het bereik van de CSS tot alleen een bepaald onderdeel van de pagina (*namespacing*). Aan goed gekozen klassennamen is ook snel te zien om welke onderdeel het gaat. Neem de twee menu's uit het vorige voorbeeld:

```
<nav class="sitemenu">  
<nav class="paginamenu">
```

De onderdelen kunnen namen krijgen als:

```
<ul class="sitemenu-lijst">  
<li class="sitemenu-item">  
<a class="sitemenu-link">  
<a class="sitemenu-link-actief">
```

Op deze manier wordt nog steeds maar één klasse gebruikt (lage specificiteit), terwijl de onderdelen makkelijk zijn te selecteren.

Van alles wat

Klassen combineren met elementnamen is ook een mogelijkheid. Als meerdere onderdelen in de pagina bestaan uit hetzelfde element, kan met een klasse heel goed worden aangeduid om welke element het gaat. Dat werd hiervoor gedaan met de menu's:

```
<nav class="sitemenu">  
<nav class="paginamenu">
```

De onderdelen van het menu hoeven geen klassen te zijn, met elementselectors kan het ook:

```
.sitemenu { ... }  
.sitemenu ul { ... }  
.sitemenu li { ... }  
.sitemenu a { ... }
```

Het submenu kan ook een klasse krijgen om de selectoren te vereenvoudigen:

```
.sitemenu-sub ul { ... }  
.sitemenu-sub li { ... }  
.sitemenu-sub a
```

Deze laatste voorbeelden zijn wel specieker dan die met alleen klassen.

En de winnaar is...

Verwacht hier geen sluitend antwoord op de vraag wat de beste manier is om selectors te gebruiken, simpelweg omdat er niet één werkwijze is waarover iedereen het eens is. Feit is wel dat steeds vaker klassen worden gebruikt en dat daarbij een BEM-achtige naamgevingsmethode favoriet lijkt. BEM staat voor Block Element Modifier en uitleg is te vinden op getbem.com.

Kies in elke geval een methode die past bij u en bij het project; grote projecten stellen andere eisen dan kleine. Gebruikt u een CSS-bibliotheek, dan is de keus al voor u gemaakt. Uiteindelijk gaat het erom dat de code overzichtelijk, begrijpelijk en onderhoudbaar is, en dat u niet verstrikt raakt in een web van te specifieke stijlregels.

Het opbouwen van de CSS is een veelbesproken onderwerp waarover de meningen van het ene uiterste naar het andere gaan. Aanbevolen leesvoer:

- www.smashingmagazine.com/2012/06/classes-where-were-going-we-dont-need-classes/
- www.smashingmagazine.com/2012/04/decoupling-html-from-css/
- markdotto.com/2012/03/02/stop-the-cascade/
- www.smashingmagazine.com/2018/06/bem-for-beginners/
- www.smashingmagazine.com/2011/12/an-introduction-to-object-oriented-css-oocss/
- frontstuff.io/in-defense-of-utility-first-CSS
- css-tricks.com/combining-the-powers-of-sem-and-bio-for-improving-css/



Selectors level 3 en 4

De huidige selectors worden beschreven in de specificatie op www.w3.org/TR/selectors-3/. Kijk voor de toekomst naar het concept voor de volgende versie op www.w3.org/TR/selectors-4/.

Basisselectors

Basisselectors (*simple selectors*) zijn eenvoudige, enkelvoudige selectors:

- type selector
- universele selector
- klasseselector
- ID-selector
- attribuutselector
- pseudoklasse

En de winnaar is...

Verwacht hier geen sluitend antwoord op de vraag wat de beste manier is om selectors te gebruiken, simpelweg omdat er niet één werkwijze is waarover iedereen het eens is. Feit is wel dat steeds vaker klassen worden gebruikt en dat daarbij een BEM-achtige naamgevingsmethode favoriet lijkt. BEM staat voor Block Element Modifier en uitleg is te vinden op getbem.com.

Kies in elke geval een methode die past bij u en bij het project; grote projecten stellen andere eisen dan kleine. Gebruikt u een CSS-bibliotheek, dan is de keus al voor u gemaakt. Uiteindelijk gaat het erom dat de code overzichtelijk, begrijpelijk en onderhoudbaar is, en dat u niet verstrikt raakt in een web van te specifieke stijlregels.

Het opbouwen van de CSS is een veelbesproken onderwerp waarover de meningen van het ene uiterste naar het andere gaan. Aanbevolen leesvoer:

- www.smashingmagazine.com/2012/06/classes-where-were-going-we-dont-need-classes/
- www.smashingmagazine.com/2012/04/decoupling-html-from-css/
- markdotto.com/2012/03/02/stop-the-cascade/
- www.smashingmagazine.com/2018/06/bem-for-beginners/
- www.smashingmagazine.com/2011/12/an-introduction-to-object-oriented-css-oocss/
- frontstuff.io/in-defense-of-utility-first-CSS
- css-tricks.com/combining-the-powers-of-sem-and-blo-for-improving-css/



Selectors level 3 en 4

De huidige selectors worden beschreven in de specificatie op www.w3.org/TR/selectors-3/. Kijk voor de toekomst naar het concept voor de volgende versie op www.w3.org/TR/selectors-4/.

Basisselectors

Basisselectors (*simple selectors*) zijn eenvoudige, enkelvoudige selectors:

- type selector
- universele selector
- klasseselector
- ID-selector
- attribuutselector
- pseudoklasse

```
* {
  color: blue;
}
```

Blauwe tekst is misschien niet zo'n voor de hand liggende toepassing, maar een praktisch voorbeeld zit in een veel toegepaste techniek:

```
html {
  box-sizing: border-box;
}
*, *::before, *::after {
  box-sizing: inherit;
}
```

De eigenschap `box-sizing` komt later aan de orde, net als de pseudo-elementen `::before` en `::after`. Hier gaat het erom dat met de universele selector een eigenschap wordt ingesteld op elk element in het document.

Er zijn meer mogelijkheden. Bekijk het volgende fragment:

```
<aside>
  <h3>Vragen</h3>
  <p> ...</p>
  <h3>Meer vragen</h3>
  <p> ... </p>
  <h3>Eindelijk antwoorden</h3>
  <p> ... </p>
</aside>
```

En bekijk de volgende CSS-regel:

```
h3 {
  color: blue;
}

* + h3 {
  color: red;
}
```

De selector `* + h3` vindt elke `<h3>` die op hetzelfde niveau staat als het voor-gaande element (*sibling*). De eerste `<h3>` wordt niet voorafgegaan door een element op hetzelfde niveau en wordt daarom blauw; de andere worden rood.

Klasseselector

Het gebruik van de klasseselector is in hiervoor besproken, maar hoe een klasse wordt ingesteld is nog niet uitgelegd. Stel bij het HTML-element het attribuut `class` in. In de CSS wordt naar de klassenaam verwezen door voor die naam een punt te zetten:

```
<p class="intro">

.intro {
    line-height: 1.5;
}
```

Een element kan tot meer klassen behoren. Klassenamen bij het attribuut `class` worden door een spatie van elkaar gescheiden.

```
<button class="knop verzend">

.knop {
    algemene opmaak van een knop
}

.verzend {
    specifieke opmaak verzendknop
}
```

Kenmerken van een klasse:

- De naam van een klasse wordt voorafgegaan door een punt (.). Het HTML-element `<li class="sitemenu-item">` wordt in CSS-code `.sitemenu-item { ... }`.
- Klassen kunnen meermaals worden gebruikt op dezelfde of verschillende elementen in een HTML-document.
- Namen van klassen zijn in HTML 5 niet hoofdlettergevoelig, maar in HTML 4 wel. Om problemen te voorkomen worden daarom klassen (en ID's) altijd gebruikt alsof ze hoofdlettergevoelig zijn. De browsers behandelen klassen (en ID's) ook hoofdlettergevoelig: `mijnKlasse` is dus niet hetzelfde als `mijnklasse`.
- Een naam mag bestaan uit elke combinatie van tekens, dus letters, cijfers, streepjes enzovoort, maar er mag geen spatie in staan en de naam mag niet beginnen met een cijfer. Aanbevolen wordt een naam te kiezen die de aard van de inhoud beschrijft en niet de gewenste opmaak.

ID-selector

De ID-selector is voor CSS een beetje een buitenbeentje geworden. Voor HTML5 werd deze selector vooral gebruikt om kenmerkende, unieke secties te markeren, zoals header, footer, nav enzovoort. In die zin is de ID voor opmaak overbodig. Wel hebt u eerder gezien dat de ID in HTML wordt gebruikt om een `<label>` en een `<input>` te koppelen of dient als anker voor een koppeling naar een fragment in een pagina (bladwijzer). Verder wordt de ID gebruikt als haakje voor JavaScript en een contentmanagementsystemen zoals WordPress leunen ook stevig op de ID.

Voor CSS heeft de ID een unieke eigenschap: hij is uniek. Een ID mag maar één keer voorkomen en daar kunt u zeer specifieke opmaak aan koppelen. Een ID is behalve uniek ook zeer specifiek, dus een opmaakeigenschap wordt niet snel overgeschreven. Afhankelijk van de toepassing is dat een voordeel of een nadeel, maar zolang u zich daar bewust van bent, hoeft dat geen probleem te zijn.

In de HTML gebruikt u het attribuut `id`. In de CSS krijgt die naam een hekje ervoor om er een selector van te maken:

```
<header id="paginaheader">

#paginaheader {
    width: 100vw;
}
```

Kenmerken van de ID:

- De naam van een ID wordt voorafgegaan door een hekje (#). Het HTML-element `<div id="container">` wordt in CSS-code `#container { ... }`.
- Een ID mag in een document maar één keer voorkomen.
- Namen van ID's zijn hoofdlettergevoelig. Alles wat geldt voor de namen van klassen, geldt ook voor de namen van ID's.
- Een naam mag bestaan uit elke combinatie van tekens, dus letters, cijfers, streepjes enzovoort, maar er mag geen spatie in staan en de naam mag niet beginnen met een cijfer. Dat zijn dezelfde eisen als bij een klasse.

Attribuutselectors

Met attribuutselectors selecteert u elementen die een bepaald attribuut hebben of waarvan een attribuut een bepaalde waarde heeft.

- `img[alt]` Elk element `` met het attribuut `alt`, ongeacht de waarde ervan.

- `img[alt="foto"]` Elk element `` waarvan het attribuut `alt` precies overeenkomt met de waarde foto. Alle afbeeldingen waarvan de alt-tekst leeg is, vindt u met `img[alt=""]`.
- `img[alt^="foto"]` Elk element `` waarvan het attribuut `alt` begint met de waarde foto, bijvoorbeeld fotograaf.
- `img[alt$="foto"]` Elk element `` waarvan het attribuut `alt` eindigt met de waarde foto, bijvoorbeeld luchtfoto.
- `img[alt*="foto"]` Elk element `` waarvan het attribuut `alt` in elke geval de waarde foto bevat. Dus ook fotoboek of Een foto van de tuin.
- `img[class+="fotoboek"]` Elk element `` waarvan het attribuut `class` meerdere waarden kan hebben, waaronder de waarde fotoboek.
- `a[hreflang]="nl"` Elk element `<a>` waarvan het attribuut `hreflang` de waarde nl heeft of begint met nl, direct gevolgd door een koppelteken. Komt bijvoorbeeld overeen met de taalcodes nl, nl-NL en nl-BE.

Pseudoklassen

In CSS zijn er twee typen pseudoselectors: pseudoklassen en pseudo-elementen. Vrij vertaald geeft een pseudoklasse een toestand van een element aan, bijvoorbeeld aangewezen met de muis (`:hover`) of geselecteerd voor invoer (`:focus`). Het andere type is pseudo-element en dat heeft die naam omdat het gezien zou kunnen worden als een zelfstandig element. Een voorbeeld is `::before`. Pseudo-elementen zijn een zelfstandige groep en vallen niet onder de basisselectors.

Dynamische pseudoklassen

In deze categorie vallen kenmerken van elementen die niet in code zichtbaar worden; de documentstructuur verandert er niet door.

- `:link` Een niet-bezochte koppeling,
- `:visited` Een bezochte koppeling.

De pseudoklasse `:visited` is om privacyredenen beperkt in de mogelijkheden voor opmaak. De achtergrond is dat voorheen via kwaadaardige websites het surfgedrag kon worden achterhaald aan de hand van typerende opmaak van bezochte links. Dat gebeurt natuurlijk nog steeds via tracking cookies, maar daar hebt u nog controle over. De enige eigenschappen van `:visited` die kunnen worden aangepast zijn:

- `color`
- `background-color`
- `border-color` (en subeigenschappen)
- `column-rule-color`
- `outline-color`

Het is bij voorbeeld niet mogelijk om de onderstreping van alleen de bezochte links uit te schakelen, maar wel die van alle links. Voor aangepaste onderstreping gebruikt u bij voorbeeld de onderste rand:

```
a:link {
    text-decoration-line: none;
    border-bottom: 1px solid red;
    color: black;
}

a:visited {
    border-bottom: 1px solid white;
    color: inherit;
}
```



Vaste volgorde

Om te voorkomen dat andere pseudoklassen de opmaak van :link niet kunnen overschrijven, moeten de stijlregels een vaste volgorde hebben: :link, :visited, :hover, :active.

Handeling van de gebruiker

- :hover Een element waar de muis boven zweeft.
- :active Een actief element, bijvoorbeeld een koppeling waarop wordt geklikt. Let op: het gaat om het moment van klikken; zodra de muisknop wordt losgelaten is :active niet meer van toepassing.



Klikeffect op mobiel

Een aanraakscherm herkent niet :hover, want er is geen muis waarmee iets kan worden aangewezen. Een aanwijseffect is dus nutteloos op een mobiel. Tikken wordt uiteraard wel herkend en dat activeert de pseudoklasse :active. Een klikeffect kan dus worden gemaakt met de selector :active (Denk aan een kleurverandering van een knop.) Dat werkt als klikeffect uiteraard ook op een gewoon scherm.

-
- :focus Een element dat de focus heeft en invoer kan ontvangen, bijvoorbeeld <input>.
 - Pseudoklasse doel
 - :target Dit element is het doel van een link naar een bepaalde locatie in het document (het element heeft een ID en de verwijzende URL eindigt op #id-naam, zie ook hoofdstuk 5, de paragraaf *Bladwijzers maken*). Met :target kunt u dergelijke doelelementen opmaken.

```
<!--dit is de link-->
<a href="www.domain.nl/pagina.html#doel">Naar het doel</a>
<!-- dit is het doel -->
<h1 id="doel">Hier landt de link</h1>
/* dit is de CSS */
:target { ... } /* elk doel in een document */
h1:target { ... } /* alleen h1-elementen die een doel zijn */
```

Pseudoklasse taal

- :lang() Een element waarvan de taal overeenkomt met die in de selector.

```
<div lang="en">
div:lang(en) { ... }
```

Pseudoklasse UI-toestand

- :enabled Elementen in de gebruikersinterface die beschikbaar zijn.
- :disabled Elementen in de gebruikersinterface die niet beschikbaar zijn.
Het gaat hier om invoerelementen en knoppen.
- :checked Een keuzerondje of selectievakje dat is ingeschakeld.

Structurele pseudoklassen

- :root Het hoofdelement van het document, de root, in HTML altijd `<html>`.
- :nth-child() Selecteert elk zoveelste kindelement. Bijvoorbeeld te gebruiken om eenvoudig tabellen van wisselende kleuren te voorzien. De opbouw is `:nth-child(an+b)`. Vermenigvuldig `a` met `n` (een tel van 0 tot het aantal kindelementen) en tel dat op bij `b`. De waarde kan ook odd (oneven) of even zijn.

```
tr:nth-child(2n+1) /* alle oneven rijen in een tabel: 2x0+1=1; 2x1+1=3; 2x2+1=5; 2x3+1=7 enz. */
tr:nth-child(odd) /* hetzelfde resultaat */
tr:nth-child(2n+0) /* alle even rijen in een tabel: 2x0+0=0; 2x1+0=2; 2x2+0=4; 2x3+0=6 enz. */
tr:nth-child(2n) /* hetzelfde resultaat, b=0 mag worden weggelaten */
tr:nth-child(even) /* hetzelfde resultaat */
```

- :nth-last-child() Hetzelfde als de voorgaande, maar gerekend vanaf het laatste kind. In het volgende voorbeeld wordt eerst de tweede rij van onder geselecteerd ($-1 \times 0 + 2 = 2$), daarna wordt de laatste geselecteerd ($-1 \times 1 + 2 = 1$). Als `a = 1` of `-1` mag het getal worden weggelaten.

```
tr:nth-last-child(-n+2) /* de laatste 2 rijen van een tabel */
```

Tabel met zebrastrepen	

Afbeelding 8.7 Elke oneven rij in deze tabel heeft een achtergrondkleur.

Tabel met afgezakte zebrastrepen	

Afbeelding 8.8 Alleen de laatste twee rijen zijn geselecteerd.

- `:nth-of-type()` Elk zoveelste element van een type. De berekening gaat zoals bij `:nth-child()`.
- `:nth-last-of-type()` Hetzelfde als de voorgaande, maar gerekend vanaf het laatste element.
- `:first-child` Een element dat het eerste kind is van zijn ouderelement. Dit is hetzelfde als `:nth-child(1)`.
- `:last-child` Een element dat het laatste kind is van zijn ouderelement. Dit is hetzelfde als `:nth-last-child(1)`.
- `:first-of-type` Het eerste element van een type. Dit is hetzelfde als `:nth-of-type(1)`.

```
p:first-of-type /* de eerste alinea na een ouderelement, bijvoorbeeld een kop */
```

- `:last-of-type` Het laatste element van een type, gelijk aan `:nth-last-of-type`.
- `:only-child` Een element waarvan de ouder geen andere kindelementen heeft. Kan ook worden aangegeven met `:first-child:last-child` of `:nth-first-child(1):nth-last-child(1)`. Het verschil zit in de specifi-

- citeit: de laatste twee bestaan uit twee pseudoklassen en dan is de specificiteit hoger.
- * :only-of-type Het enige kind van het bepaalde type. Komt verder volledig overeen met :only-child, ook wat betreft specificiteit.
- * :empty Selecteert een volledig leeg element. <p></p> komt overeen met p:empty.

Pseudoklasse ontkenning

- * :not(x) Selecteert elementen die niet beschikken over het tussen haakjes genoemde. De volgende declaratie markeert alle -elementen die geen attribuut alt hebben:

```
img:not([alt])
```

Met :not() kunnen alleen basiselectors worden gebruikt. Dat betekent dat bijvoorbeeld het volgende *niet* mogelijk is:

```
input:not(::first-line) { ... }
```

De selector ::first-line is namelijk geen basiselector maar een pseudo-element.

Pseudo-elementen

Pseudo-elementen bieden toegang tot onderdelen van het document waarvoor geen gewone elementen bestaan, bijvoorbeeld de eerste regel of de eerste letter van een alinea. Er kan ook inhoud worden toegevoegd voor of achter een element. Waar pseudoklassen beginnen met een enkele dubbelepunt (:), beginnen pseudo-elementen met een dubbele dubbelepunt (::).



Twee dubbelepunten is nieuw

De notatie met twee dubbelepunten is geïntroduceerd in Selectors level 3. In oudere versies van CSS werd ook bij pseudo-elementen één dubbele punt gebruikt. Alleen voor die pseudo-elementen blijven browsers de notatie met één dubbele punt ondersteunen. Dat zijn :first-line, :first-letter, :before en :after. Alle toekomstige pseudo-elementen kunnen alleen met twee dubbelepunten worden geschreven..

- * ::first-line De eerste regel opgemaakte tekst van een element. Let op: het is niet uw eerste zin, maar de eerste regel zoals die in de browser te zien is. Niet alle opmaakeigenschappen zijn beschikbaar, maar in geval

citeit: de laatste twee bestaan uit twee pseudoklassen en dan is de specificiteit hoger.

- `:only-of-type` Het enige kind van het bepaalde type. Komt verder volledig overeen met `:only-child`, ook wat betreft specificiteit.
- `:empty` Selecteert een volledig leeg element. `<p></p>` komt overeen met `p:empty`.

Pseudoklasse ontkenning

- `:not(x)` Selecteert elementen die niet beschikken over het tussen haakjes genoemde. De volgende declaratie markeert alle ``-elementen die geen attribuut `alt` hebben:

```
img:not([alt])
```

Met `:not()` kunnen alleen basiselectors worden gebruikt. Dat betekent dat bijvoorbeeld het volgende *niet* mogelijk is:

```
input:not(::first-line) { ... }
```

De selector `::first-line` is namelijk geen basiselector maar een pseudo-element.

Pseudo-elementen

Pseudo-elementen bieden toegang tot onderdelen van het document waarvoor geen gewone elementen bestaan, bijvoorbeeld de eerste regel of de eerste letter van een alinea. Er kan ook inhoud worden toegevoegd voor of achter een element. Waar pseudoklassen beginnen met een enkele dubbelepunt (`:`), beginnen pseudo-elementen met een dubbele dubbelepunt (`::`).



Twee dubbelepunten is nieuw

De notatie met twee dubbelepunten is geïntroduceerd in *Selectors level 3*. In oudere versies van CSS werd ook bij pseudo-elementen één dubbele punt gebruikt. Alleen voor die pseudo-elementen blijven browsers de notatie met één dubbele punt ondersteunen. Dat zijn `:first-line`, `:first-letter`, `:before` en `:after`. Alle toekomstige pseudo-elementen kunnen alleen met twee dubbelepunten worden geschreven.

- `::first-line` De eerste regel opgemaakte tekst van een element. Let op: het is niet uw eerste zin, maar de eerste regel zoals die in de browser te zien is. Niet alle opmaakeigenschappen zijn beschikbaar, maar in geval

wel alles wat betrekking heeft op tekst, lettertype en (achtergrond)kleur. Met de volgende code wordt de eerste regel van een bericht vet gemaakt:

```
p::first-line { font-weight: 700; }
```

Van deze tekst is de eerste regel vet, doordat de pseudoklasse :first-line is toegepast. De eerste regel is niet hetzelfde als de eerste zin.

Van deze tekst is de eerste regel vet, doordat de pseudoklasse :first-line is toegepast. De eerste regel is niet hetzelfde als de eerste zin.

Afbeelding 8.9 Het effect van ::first-line (08_09.html).

- **::first-letter** De eerste letter. Een klassieker, in alle opzichten (denk aan de fraai versierde letters in heel oude boeken: monnikenwerk). Er kan een initiaal (*drop cap*) mee worden gemaakt, een extra grote eerste letter, maar bij getallen werkt het ook. Een leesteken en een letter, bijvoorbeeld als de tekst begint met een aanhalingsteken, vallen gewoonlijk beide onder ::first-letter.

```
p::first-letter { font-size: 300%; }
```

De eerste letter heeft een grootte van 300%. Het is nu een initiaalletter geworden. Het bijeffect is dat de regelhoogte van de eerste regel groter is dan die van de andere regels. Dat verschil wordt gecompenseerd met line-height op de :first-line.

De eerste letter heeft een grootte van 300%. Het is nu een initiaalletter geworden. Het bijeffect is dat de regelhoogte van de eerste regel groter is dan die van de andere regels. Dat verschil wordt gecompenseerd met line-height op de :first-line.

Afbeelding 8.10 De initiaalletter, met leesteken (08_10.html).

- **::before** Tekst invoegen voor de inhoud van het element.
 - **::after** Tekst invoegen na de inhoud van het element.
- Met deze pseudo-elementen is veel mogelijk, van automatische nummering (hoofdstukken, paragrafen, afbeeldingen) tot speciale pictogrammen en alles wat u zelf nog kunt verzinnen. De eigenschap content specificert de inhoud.
- U begrijpt nu waarschijnlijk waarom het een pseudo-element wordt genoemd: er wordt in feite extra HTML toegevoegd. In de ontwikkelaars-hulpmiddelen staan ::before en ::after in de HTML-boomstructuur. De inhoud ervan en hoe die eruitziet staat bij de CSS-weergave.

```

/* automatisch genummerde figuren */
body {
    counter-reset: figuren; /* zet de teller figuren op 0 */
}
figcaption::before {
    content: "Figuur " counter(figuren) " "; /* voegt toe: Figuur x en een spatie na het nummer */
    counter-increment: figuren; /* verhoog de teller figuren met 1 */
}
/* een pictogram voor een opmerking, zoals in dit boek */
p.note::before {
    content: url(tip.png);
    float: left;
    margin-right: 4px;
}
/* een pijl achter een link naar een andere site */
a[href^="http://"]::after { /* het attribuut href begint met http:// */
    content: url(Icons/pijl.png);
}

```



Phasellus quis odio aliquam, pharetra eros accumsan, tempor sapien. Aliquam in volutpat nunc, ac laoreet nulla. Integer luctus tellus quis orci mollis tempor. Suspendisse laoreet nulla sem, vel commodo risus feugiat vitae. Morbi pulvinar mattis quam, et dignissim est volutpat eget. Etiam dapibus sapien in urna lobortis facilisis.

Afbeelding 8.11 Pictogram voor de tekst met ::before.

Pseudo-elementen level 4

Nieuwe pseudo-elementen staan niet in Selectors level 4, maar in de nieuwe module *Pseudo-elements level 4* (www.w3.org/TR/css-pseudo-4/). Ze worden nog lang allemaal ondersteund door de browsers, maar twee ervan zijn bruikbaar.

- **::selection** Door de gebruiker geselecteerde tekst op de pagina of in een tekstvak (input, textarea).
- **::placeholder** Plaatshoudertekst in bijvoorbeeld een invoerveld. Voor de opmaak van plaatshoudertekst gelden dezelfde beperkingen als voor ::first-line.

Andere (nog niet-ondersteunde) pseudo-elementen zijn:

- **::selection-inactive** Selectie in een niet actief (browser)venster.
- **::spelling-error** Tekst die volgens de browser verkeerd is gespeld. Dit en het volgende element zijn vooral nuttig bij tekstinvoervelden.
- **::grammar-error** Tekst die volgens de browser grammaticaal fout is.

- ::marker Het blok dat de markering van een lijselement omvat, de *marker box*. Geldige eigenschappen zijn `font-*` en `color`.

Combinatieselectors: afstamming, kind en sibling

Geen combinator: komma

Het belangrijke verschil tussen een spatie en een komma bij selectoren is al genoemd. Ter herinnering: de komma is het scheidingsteken tussen de selectoren. De volgende regel selecteert alle genoemde elementen op elke plek in het document:

```
h1, p, ul, li { ... }
```

Algemene afstamming: spatie

De spatie geeft algemene afstamming aan. Het gewenste element kan een kind zijn, maar een kleinkind of een achterkleinkind voldoet ook aan de voorwaarde van deze algemene afstammingscombinator. In de volgende regel heeft de declaratie betrekking op elke `` die afstamt van `<main>`:

```
main em {
  font-style: italic;
  color: green;
}
```

Het maakt niet uit waar deze `` staat, als het maar binnen het element `<main>` is. In de volgende code wordt elke `` cursief en groen. (Deze code wordt voor alle volgende voorbeelden gebruikt.)

```
<main>
  <article>
    <h2>Artikelkop</h2>
    <p>Lorem ipsum. <em>A, nobis</em> sit quod atque.</p>
    <ul>
      <li>Lorem ipsum dolor sit amet,</li>
      <li>Quod <em>distinctio</em> atque ipsum illio minus,</li>
      <li>Dolorum reprehenderit suscipit eligendi.</li>
    </ul>
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.</p>
    <h3>Tussenkop</h3>
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit.</p>
  </article>
</main>
```

Kindselector

Kindelementen worden geselecteerd met de punthaak > (de kindcombinator oftewel kindselector). Toegepast op het HTML-voorbeeld wordt alleen de in de <p> geselecteerd met:

```
p > em {  
    color: purple;  
}
```

Merk op dat alle elementen cursief blijven. Alleen de eigenschap die opnieuw wordt ingesteld verandert.

Aangrenzend: +

Het plusje selecteert elementen op hetzelfde niveau die direct worden voorafgegaan door het andere geroemde element, bijvoorbeeld een die direct wordt voorafgegaan door . Hiermee wordt elk item in een lijst geselecteerd, behalve het eerste (want daar staat geen voor, het is het eerste kind van):

```
li + li {  
    background-color: chocolate;  
}
```

Het eerste item kan niet worden gevonden met (onzinalert!) li:not(+li) of iets dergelijks, omdat bij :not() geen combinators kunnen worden gebruikt, alleen basiselectors. Het eerste item in een lijst wordt wel gevonden met de pseudoklasse li:first-child of li:first-of-type.

Op hetzelfde niveau: ~

General sibling (of subsequent sibling in Selectors level 4) is een lastig te vertalen term die hoort bij de combinator tilde (~). ‘Op hetzelfde niveau’ komt nog het dichtst in de buurt. Het gaat om elementen op hetzelfde niveau (ze hebben dus dezelfde ouder), die staan binnen het eerstgenoemde element.

U selecteert als volgt elke <p> die komt na , waarbij beide elementen afstammen van dezelfde ouder (hier <article>):

```
ul ~ p {  
    text-shadow: 1px 1px 2px black;  
}
```

De eerste <p> krijgt geen tekstschaduw, want die staat voor de .

Toegepast op de lijstitems is het resultaat hetzelfde als `li + li`. Het belangrijke verschil tussen de selectoren `+` en `-` is dat er bij `-` andere elementen tussen mogen staan. De elementen `<p>` voor en na de tussenkop `<h3>` zijn beide geselecteerd, omdat ze kinderen zijn van `<article>` en omdat ze na `` komen.



Selectors level 4

De bruikbare selectoren zijn besproken. De ontwikkelingen staan echter niet stil en er zijn meer mogelijkheden in aantocht. Het concept van *Selectors level 4* bevat ook nieuwe selectoren (die begin 2019 nog niet bruikbaar zijn). De specificatie is te vinden op www.w3.org/TR/selectors4/. Kijk op caniuse.com om te achterhalen of een eigenschap bruikbaar is.

Best practices

Omdat er geen voorschriften zijn, is het werken met selectoren een boeiende bezigheid. Gezond verstand en inzichten van anderen, online ruimschoots aanwezig, brengen u een heel eind. Bedrijven hebben vaak eigen regels. Die van Google zijn te vinden op github.com/google/styleguide en de HTML/CSS-gids staat op google.github.io/styleguide/htmlcssguide.html. GitHub heeft een stijlgids op styleguide.github.com/primer/. Deze is gemaakt voor het frontendframework Primer, dat wordt gebruikt op Github. Ook interessant is de stijlgids van WordPress, zelfs als u het cms niet gebruikt. Kijk op make.wordpress.org/core/handbook/best-practices/coding-standards/css/. Daar staat ook een link naar github.com/necolas/idiomatic-css, een CSS-stijlgids geschreven door de auteur van `normalize.css`.

Het lijkt een liedje in de herhaalstand, maar goed werkende, geldige en leesbare CSS begint met betekenisvolle HTML. Als u de HTML op orde hebt, bent u al halfweg. Nog wat algemene vuistregels:

- Begin met een gelijk speelveld. Gebruik een vorm van normalisatie om browserverschillen in basisopmaak glad te strijken.
- Bouw een stylesheet op in blokken. Begin bijvoorbeeld met algemene instellingen voor de typografie, zoals lettertype, tekstgrootte en regelregelafstand. Maak blokken met stijlregels voor de hoofdonderdelen van de pagina, zoals de paginaheader, de navigatie, en de footer, maar ook voor artikelen en andere terugkerende elementen. Maak ook een blok voor de basislay-out, bijvoorbeeld een raster (*grid*).
- Kies een systeem voor de volgorde van stijlregels binnen een declaratie. Alfabetisch is prima, maar het kan ook op basis van gerelateerde eigen-

schappen. WordPress bijvoorbeeld deelt eigenschappen in op display, positionering, boxmodel, kleur en typografie, en overig.

- Het lastigst is het kiezen van een methode voor het werken met selectoren. De keus hangt af van persoonlijke voorkeur en zeker ook de aard en de omvang van het project. De mogelijkheden zijn globaal beschreven in de paragraaf *Elementnaam, klasse of beide?*
- Documenteer uw CSS-code, zodat u of iemand anders ook later nog begrijpt hoe die in elkaar zit.
- Werk DRY: don't repeat yourself. Voorkom dubbele declaraties, met andere woorden, specificeer niet op twintig plaatsen hetzelfde lettertype of een identieke kleur. Houd rekening met de cascade en met overerving. Gebruik een hulpklasse als dat nuttig is. Denk ook aan afstammingselectors.

```
23 /* ...
24 >>> TABLE OF CONTENTS:
25 -----
26 # Normalize
27 # Typography
28 # Elements
29 # Forms
30 # Navigation
31   ## links
32   ## Menus
33 # Accessibility
34 # Alignments
35 # Clearings
36 # Widgets
37 # Content
38   ## Posts and pages
39   ## Comments
40 # Infinite scroll
41 # Media
42   ## captions
43   ## galleries
44 -----
45 */
```

Afbeelding 8.12 Een voorbeeld van groepering in een stylesheet. Hier een fragment van een WordPress-thema.

Samenvatting

In dit hoofdstuk is de basis gelegd voor het opmaken van HTML-pagina's met CSS. Besproken zijn onder meer:

- De achtergronden en voordelen van CSS.
- Hoe de browser CSS verwerkt en waarom die kennis belangrijk is, bijvoorbeeld in relatie tot specificiteit.
- Hoe specificiteit wordt berekend en waarom u moet oppassen met !important.

- Hoe stijlregels worden geschreven en welke waarden en eenheden er zijn.
- Waar de stijlregels worden geplaatst en hoe ernaar wordt verwezen.
- Welke selectoren er zijn en hoe ze worden gebruikt om elementen in het DOM te benaderen en manipuleren.

Het volgende hoofdstuk gaat over twee andere kernonderdelen van CSS: het boxmodel en het weergavemodel. U leert hoe een pagina vanuit de HTML en CSS wordt opgebouwd en hoe u daar invloed op kunt uitoefenen.

Oefeningen

- Bezoek verschillende websites en onderzoek met behulp van de Hulpmiddelen voor ontwikkelaars de HTML- en CSS-code. Kunt u achterhalen hoe de verschillende soorten selectoren zijn gebruikt? Bekijk ook wat het effect is als u stijlregels uitschakelt of verandert; welke elementen worden door een stijlregel beïnvloed?



Stijlregels aanpassen

In bijvoorbeeld Chrome DevTools kiest u de optie Sources. Zoek in de mappenlijst bestanden met de extensie .css en klik daarop. In het rechterdeelvenster verschijnt de CSS-code. Deze kunt u aanpassen het resultaat daarvan is direct in de pagina te zien.

- Maak een HTML-pagina en probeer met behulp van de diverse selectoren elk onderdeel een ander CSS-opmaakkenmerk te geven. Het doel is het leren werken met selectoren. De opmaak is alleen ter controle, dus een andere tekstkleur, achtergrondkleur of een rand is voldoende.
 - Plaats de CSS-code ofwel in het attribuut `<style>` van het HTML-bestand of maak een apart bestand stijlen.css en `<link>` daarnaar vanuit het HTML-bestand. Beide methoden zijn in dit hoofdstuk beschreven.
 - De eigenschap voor tekstkleur is `color`, achtergrondkleur is `background-color`. Gebruik bijvoorbeeld grijswaarden zoals #666, #888, #aaa of #ccc. (Werken met kleur en kleurwaarden wordt uitgelegd in hoofdstuk 12.)

```
selector {
  color: #666;
}
```

- De eigenschap voor een rand is border. Met de declaratie border: 1px solid black; maakt u rondom een element een zwarte rand van 1 pixel dik. (Randen worden uitgelegd in hoofdstuk 9.)

```
selector {  
    border: 1px solid black;  
}
```



Gebruik de voorbeeldcode

Van vrijwel alle afbeeldingen is de HTML- en CSS-code beschikbaar. De bestandsnaam staat in het bijschrift. Gebruik deze code om te oefenen: pas de code aan, voeg dingen toe, speel ermee! Kijk op handboek-html-css.nl.

- De eigenschap voor een rand is `border`. Met de declaratie `border: 1px solid black;` maakt u rondom een element een zwarte rand van 1 pixel dik. (Randen worden uitgelegd in hoofdstuk 9.)

```
selector {  
    border: 1px solid black;  
}
```



Gebruik de voorbeeldcode

Van vrijwel alle afbeeldingen is de HTML- en CSS-code beschikbaar. De bestandsnaam staat in het bijschrift. Gebruik deze code om te oefenen: pas de code aan, voeg dingen toe, speel ermee! Kijk op handboek-html-css.nl.

Lay-out: boxmodel en weergavemodel

Er is nog heel wat voor nodig om van alle losse HTML-elementen een mooi opgemaakte pagina te maken, meer nog dan u misschien op het eerste gezicht denkt. Als de HTML-code is geladen, maakt de browser van elk element een box. Elke box heeft eigenschappen zoals hoogte, breedte, marge, padding en rand. Naast die eigenschappen voor het formaat en andere uiterlijke kenmerken is er ook een systeem dat het gedrag van de boxen bepaalt. Er zijn boxen die alleen kunnen worden gestapeld en andere kunnen naast elkaar in de regel staan. Dit zijn lang bestaande kerneigenschappen van CSS, maar de ontwikkelingen staan niet stil. Daarom wordt ook gekeken naar een nieuwe manier van denken over breedte en hoogte en naar de ontwikkelingen bij het uitlijnen van boxen. Al bij al geeft dit hoofdstuk de theoretische ondergrond voor het maken van lay-outs met CSS, dat in het volgende hoofdstuk wordt behandeld.

U leert in dit hoofdstuk:

De kenmerken van het boxmodel: breedte, hoogte, marge, padding en rand.

Hoe de grootte van een blok wordt bepaald door box-sizing.

Anders denken over breedte en hoogte: logische eigenschappen.

Nieuwe mogelijkheden voor uitlijning: box alignment.

Het weergavemodel en de eigenschap display.

Het (niet) verbergen van inhoud: visibility en overflow.

De opbouw van pagina's

De weergave van een niet-opgemaakte HTML-pagina komt vaak nog het meest overeen met een standaardpagina in een tekstverwerker. Alles staat netjes onder elkaar, koppen zijn groter dan gewone tekst, er is witruimte tussen koppen en alinea's, er is ingesprongen tekst, er staan bullets bij opsommingen en er zijn vette en cursieve woorden of zinsdelen. Hoe vanzelfsprekend dat ook lijkt, er zitten allerlei mechanismen achter die al die HTML-elementen de eigenschappen geven waarmee de browser met zijn ingebouwde stijlblad deze weergave maakt.

CSS is verantwoordelijk voor de opmaak, ook van dat kale maar overzichtelijke document. Wordt de pagina door de ontwerper met CSS opgemaakt, dan kan die volledig naar wens worden ingedeeld. Daarbij worden nog steeds dezelfde mechanismen gebruikt, alleen bepaalt nu de ontwerper de instellingen ervan.

In dit hoofdstuk gaat het over de basiskennis voor het maken van lay-outs met CSS. We kijken naar:

- het *boxmodel* dat van elk element een box maakt;
- het *weergavemode* dat de ordening van die boxen en de inhoud ervan bepaalt met de eigenschap `display`;
- inhoud verbergen met de eigenschap `visibility`;
- wat er gebeurt als er te veel inhoud is voor de ruimte in de box (`overflow`).

De volgende stap is het toepassen van de verschillende lay-outsysteem en dat gebeurt in het volgende hoofdstuk.

Browserstijlen zijn er niet voor niets

Hoewel ontwikkelaars de opmaakstijlen van de browser op alle mogelijke manieren kunnen aanpassen, is die standaardopmaak er natuurlijk niet voor niets. Als het om wat voor reden dan ook niet lukt om de CSS-stylesheets te laden, krijgt de gebruiker toch een leesbare, bruikbare HTML-pagina voorgesloten. Zo kan hij toch dat artikel lezen of uw product bestellen. Daarbij geldt wel een belangrijke voorwaarde: u moet een bruikbare HTML-pagina hebben gecodeerd, met betekenisvolle elementen op de juiste wijze toegepast. Dat blijft de kern van uw ontwerpen, CSS of geen CSS.

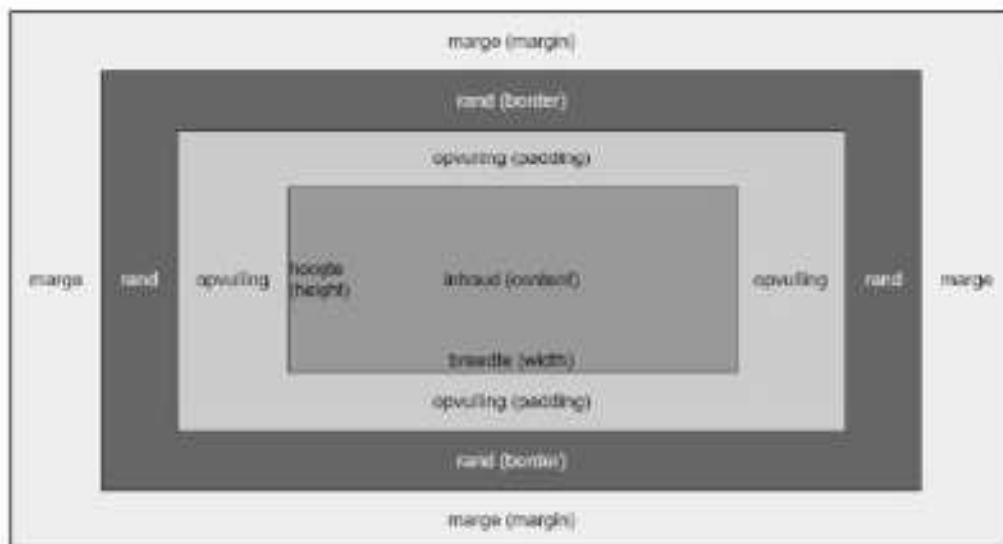
Het boxmodel

Een webpagina is opgebouwd uit boxen. Er zijn boxen met koptekst, alinea-tekst, opsommingen, tabellen, afbeeldingen enzovoort. Elk onderdeel op de pagina is een box. Het formaat en de uiterlijke eigenschappen van zo'n box worden beschreven in het boxmodel. Daarover gaat deze paragraaf.

Hoe die boxen op een pagina zich tot elkaar verhouden, met andere woorden: hoe met al die boxen de lay-out wordt opgebouwd, wordt bepaald door weergavemodel (*visual formatting model*) en de eigenschap `display`. Die eigenschap bepaalt hoe de boxen meedoen in de lay-out en hoe de inhoud van die boxen zich gedraagt. Daarmee wordt uiteindelijk de lay-out van de hele pagina opgebouwd.

Terug naar de boxmodel. De afbeelding toont schematisch hoe een box is opgebouwd:

- er is inhoud (tekst of beeld);
- er is ruimte tussen de inhoud en de rand: padding;
- er is een rand: border;
- er is ruimte aan de buitenkant: margin.



Afbeelding 9.1 Het boxmodel (09_01.html).

Niet alle onderdelen hoeven ruimte in te nemen. Inhoud kan strak tegen de rand staan en dan is de padding nul. De rand kan strak tegen het volgende blok staan en dan is de marge nul. Een marge kan zelfs negatief zijn en dan verschuift de inhoud (zie de paragraaf *Negatieve marge*). Een rand kan nul zijn, maar ook heel dik om speciale effecten te maken.

De eigenschap box-sizing

De totale ruimte die een box inneemt in de lay-out bestaat uit:

- breedte: width
- hoogte: height
- breedte/hoogte van de padding
- dikte van de rand
- breedte/hoogte van de marge

Het is wel even opletten bij het maken van boxen, want de berekende waarde van width en height hangt af van de eigenschap box-sizing. Met die eigenschap kan de ruimte voor de inhoud aanzienlijk verschillen. De mogelijke waarden zijn:

- content-box (standaardwaarde)
- border-box



Specificatie

De eigenschap box-sizing maakt deel uit van de module *CSS Basic User Interface level 3* (CSS3 UI): www.w3.org/TR/css-ui-3/, maar zou kunnen verhuizen naar de module *CSS Intrinsic & Extrinsic Sizing level 3*, zie www.w3.org/TR/css-sizing-3/.

Traditioneel zijn width en height de maten voor het inhoudsgebied van de box. Dit komt overeen met box-sizing: content-box. De totale breedte van een box wordt dan:

width + padding links/rechts + randdikte links/rechts + marge links/rechts.
(Bereken de hoogte op vergelijkbare wijze.)

Bij de waarde border-box bestaat de ruimte voor de inhoud uit width/height) min de padding en randdikte. Voor de totale boxgrootte wordt de marge bij width/height opgeteld.

De waarde van box-sizing kan dus veel invloed hebben op de boxgrootte en de ruimte voor de inhoud. Neem bijvoorbeeld deze code:

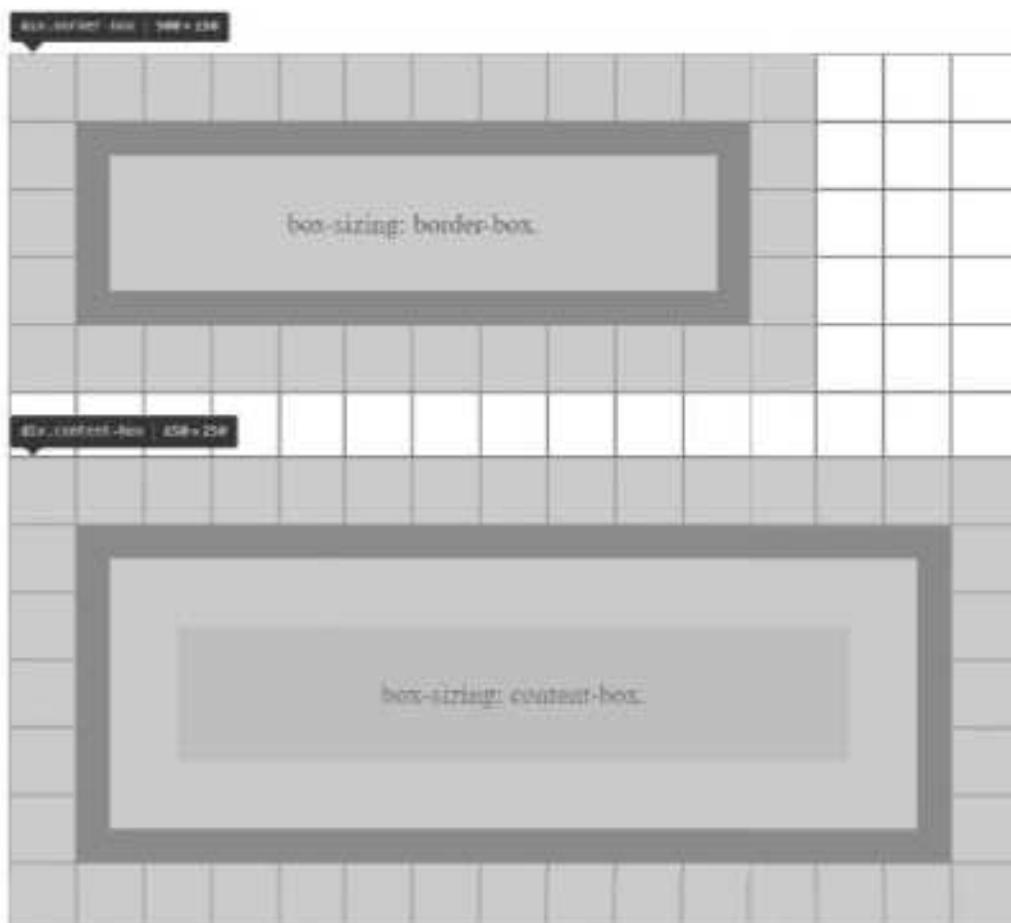
```
,border-box {  
    box-sizing: border-box; /* of content-box */  
    width: 500px;  
    height: 100px;  
    padding: 50px;  
    margin: 50px;
```

```
border: 25px solid black;
```

Bij box-sizing: content-box is de beschikbare breedte voor de inhoud 500px, namelijk de ingestelde waarde van width. De totale breedte van de box is $width: 500 + (2 \times 50 \text{ padding}) + (2 \times 25 \text{ rand}) + (2 \times 50 \text{ marge}) = 750\text{px}$.

Bij box-sizing: border-box is de breedte voor de inhoud $width: 500 - (2 \times 50 \text{ padding}) - (2 \times 25 \text{ rand}) = 350\text{px}$. De totale breedte van de hele box is $width: 500 + (2 \times 50 \text{ marge}) = 600\text{px}$.

De box met content-box is 150 pixels breder dan die met border-box en de inhoud heeft aanmerkelijk meer ruimte. In de afbeelding is het verschil goed te zien. Bedenk dat de grootte van het verschil sterk afhangt van de dikte van padding en border.



Afbeelding 9.2 Er is een groot verschil in de ruimte voor de inhoud en de uiteindelijke afmetingen tussen border-box en content-box. De breedte in de tooltip is zonder marge (09_02.html).

Nu hebben beide methoden zo hun nut. Als binnen een container de ruimte moet worden verdeeld tussen twee of meer blokken die padding en randdikte hebben, is border-box heel handig. Met een width als een percentage (én geen marge) krijgt elk blok precies het juiste deel van de beschikbare ruimte zonder rekenwerk aan padding en randdikte. Als het aantal blokken verandert, hoeft alleen het percentage te worden aangepast:

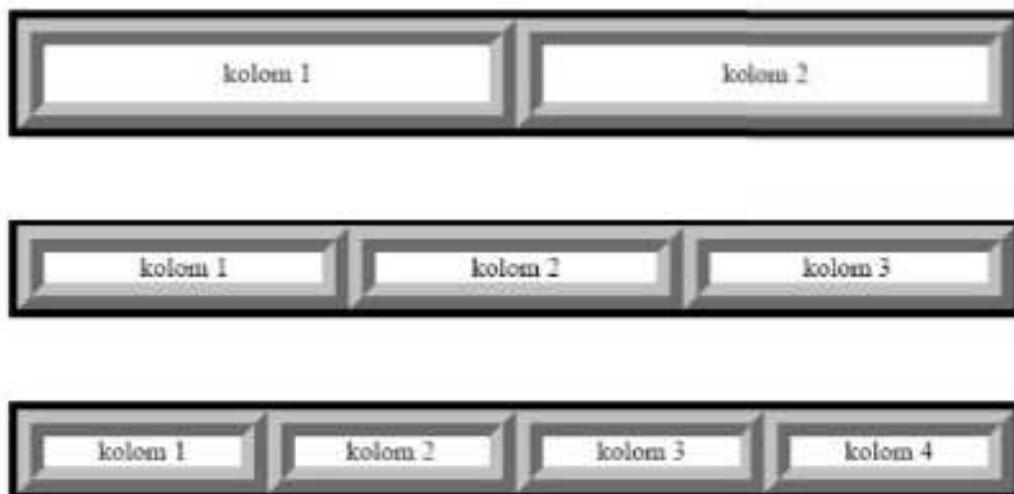
In het volgende voorbeeld is er een container van 600px breed met box-sizing: content-box. Dat betekent dat de volle 600px beschikbaar is voor de inhoud. In de container staan twee blokken met box-sizing: border-box, die elk een breedte van 50% hebben. Omdat dit de breedte inclusief de padding en de rand is, hoeft u met die twee waarden geen rekening te houden en nemen ze elk precies 300px van de breedte. Op een vergelijkbare manier kunnen er ook drie of vier blokken worden geplaatst.

```
<div class="container">
  <div class="col2">kolom 1</div>
  <div class="col2">kolom 2</div>
</div>
<div class="container">
  <div class="col3">kolom 1</div>
  <div class="col3">kolom 2</div>
  <div class="col3">kolom 3</div>
</div>
<div class="container">
  <div class="col4">kolom 1</div>
  <div class="col4">kolom 2</div>
  <div class="col4">kolom 3</div>
  <div class="col4">kolom 4</div>
</div>
/* CSS */
.container {
  box-sizing: content-box;
  width: 600px;
}
div > div {
  margin: 0;
  padding: 0.5em;
  border: 1px solid silver;
  float: left;
  text-align: center;
}
.col2 {
  box-sizing: border-box;
```

```

width: 50%;  
}  
  
.col3 {  
  box-sizing: border-box;  
  width: 33.33%;  
}  
  
.col4 {  
  box-sizing: border-box;  
  width: 25%;  
}

```



Afbeelding 9.3 Een content-box waarin de inhoud echt 600 pixels ruimte heeft, huisvest twee, drie of vier blokken met de instelling border-box ([09_03.html](#)).

Waarden voor breedte en hoogte

De breedte en hoogte worden bepaald door de eigenschappen `width` en `height`. Hiervoor zijn de volgende waarden beschikbaar:

- `auto` (standaardwaarde) de box wordt zo groot dat de inhoud erin past. De box wordt eerst zo breed als kan (begrensd door het browservenster of de container) en als dat niet genoeg ruimte geeft, wordt de box hoger. Een box kan hoger worden dan het browservenster en dan verschijnen er schuifbalken;
- `lengte` of `percentage` dit zijn de eenheden die in hoofdstuk 8 zijn besproken (de paragraaf *Waarden en eenheden*);
- `max-content` de afmeting van het grootste item in de container (bijvoorbeeld een alinea);
- `min-content` de kleinste mogelijke afmeting zonder overloop te veroorzaken, bijvoorbeeld het langste woord;

```

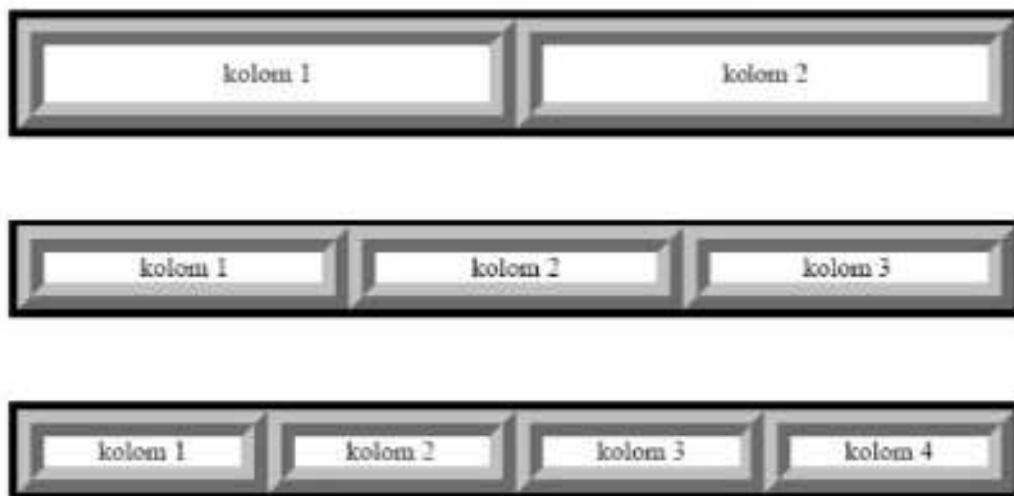
width: 50%;

}

.col3 {
  box-sizing: border-box;
  width: 33.33%;
}

.col4 {
  box-sizing: border-box;
  width: 25%;
}

```



Afbeelding 9.3 Een content-box waarin de inhoud echt 600 pixels ruimte heeft, huisvest twee, drie of vier blokken met de instelling border-box ([09_03.html](#)).

Waarden voor breedte en hoogte

De breedte en hoogte worden bepaald door de eigenschappen `width` en `height`. Hiervoor zijn de volgende waarden beschikbaar:

- `auto` (standaardwaarde) de box wordt zo groot dat de inhoud erin past. De box wordt eerst zo breed als kan (begrensd door het browservenster of de container) en als dat niet genoeg ruimte geeft, wordt de box hoger. Een box kan hoger worden dan het browservenster en dan verschijnen er schuifbalken;
- `lengte` of `percentage` dit zijn de eenheden die in hoofdstuk 8 zijn besproken (de paragraaf *Waarden en eenheden*);
- `max-content` de afmeting van het grootste item in de container (bijvoorbeeld een alinea);
- `min-content` de kleinste mogelijke afmeting zonder overloop te veroorzaken, bijvoorbeeld het langste woord;

Marges

De marge is de ruimte aan de buitenkant van een box. Een marge is altijd transparant; deze kan geen kleur of achtergrond hebben. Aangezien een box vier zijden heeft, zijn er vier marge-eigenschappen plus een verzameleigenschap:

- margin-top
- margin-right
- margin-bottom
- margin-left
- margin

Marges kunnen de gebruikelijke waarden hebben, dus pixel, em, rem, procent, plus de waarde auto.

De verzameleigenschap margin werkt op dezelfde wijze als de meeste andere verzameleigenschappen voor het boxmodel. Met de klok mee geldt:

- de eerste waarde is voor de bovenkant;
- de tweede voor de rechterkant;
- de derde voor de onderkant;
- de vierde voor de linkerkant.

Bij minder dan vier waarden betekenen deze het volgende:

- margin: 5px alle marges zijn 5 pixels breed;
- margin: 5px 3px boven en onder 5 pixels, rechts en links 3;
- margin: 5px 3px 7px boven 5, rechts 3, onder 7, links 3. Bij drie waarden wordt de vierde, ontbrekende waarde ontleend aan de tegenovergestelde kant. Links ontbreekt en wordt net als rechts 3 pixels.

Horizontaal centreren

De waarde auto betekent dat de beschikbare ruimte automatisch wordt ingevuld. Daardoor is auto erg handig voor het horizontaal centreren van boxen. Door de marges links en rechts op auto te zetten, wordt de box gecentreerd in het ouderelement (het venster of een container). Houd er wel rekening mee dat de box een waarde voor de breedte (width) moet hebben, anders werkt het niet. (Wat logisch is, want als de breedte niet bekend is, kan de vrije ruimte niet worden berekend.) In het voorbeeld zijn de marges boven en onder nul, maar die mogen elke waarde hebben.

Marges

De marge is de ruimte aan de buitenkant van een box. Een marge is altijd transparant; deze kan geen kleur of achtergrond hebben. Aangezien een box vier zijden heeft, zijn er vier marge-eigenschappen plus een verzamelleigenschap:

- margin-top
- margin-right
- margin-bottom
- margin-left
- margin

Marges kunnen de gebruikelijke waarden hebben, dus pixel, em, rem, procent, plus de waarde auto.

De verzameleigenschap margin werkt op dezelfde wijze als de meeste andere verzameleigenschappen voor het boxmodel. Met de klok mee geldt:

- de eerste waarde is voor de bovenkant;
- de tweede voor de rechterkant;
- de derde voor de onderkant;
- de vierde voor de linkerkant.

Bij minder dan vier waarden betekenen deze het volgende:

- margin: 5px alle marges zijn 5 pixels breed;
- margin: 5px 3px boven en onder 5 pixels, rechts en links 3;
- margin: 5px 3px 7px boven 5, rechts 3, onder 7, links 3. Bij drie waarden wordt de vierde, ontbrekende waarde ontleend aan de tegenovergestelde kant. Links ontbreekt en wordt net als rechts 3 pixels.

Horizontaal centreren

De waarde auto betekent dat de beschikbare ruimte automatisch wordt ingevuld. Daardoor is auto erg handig voor het horizontaal centreren van boxen. Door de marges links en rechts op auto te zetten, wordt de box gecentreerd in het ouderelement (het venster of een container). Houd er wel rekening mee dat de box een waarde voor de breedte (width) moet hebben, anders werkt het niet. (Wat logisch is, want als de breedte niet bekend is, kan de vrije ruimte niet worden berekend.) In het voorbeeld zijn de marges boven en onder nul, maar die mogen elke waarde hebben.

```
div {  
    margin-top: 0;  
    margin-right: auto;  
    margin-bottom: 0;  
    margin-left: auto;  
}
```

Met de verzameleigenschap kan het korter:

```
div {  
    margin: 0 auto;  
}
```

Een voorbeeld van een paginacontainer van 1000px breed die altijd in het midden staat:

```
<body>  
    <div class="container">  
        ... alle inhoud van de pagina ...  
    </div>  
</body>  
/* CSS */  
.container {  
    margin: 0 auto;  
    width: 1000px;  
}
```

Verticaal centreren

Een blokbox verticaal centreren is in de normale flow niet mogelijk met `margin-top: auto` en `margin-bottom: auto`, zelfs niet als de hoogte van de container bekend is. Met de nieuwe lay-outmethoden zoals flexbox en gridlayout kan het wel en nog kinderlijk eenvoudig ook. Deze methoden worden uitgelegd in hoofdstuk 10: *Lay-outs maken met CSS*.

In de normale flow is het verticaal centreren van tekst wel mogelijk, maar via een omweg. Hiervoor wordt de eigenschap `display` (zie voor meer uitleg de paragraaf *De eigenschap display*) van een container met daarin de te centreren tekst ingesteld op `table-cell`. In een tabelcel kan tekst namelijk wel verticaal worden geцentreerd. (Tekst horizontaal centreren kan altijd met `text-align: center`, zie ook hoofdstuk 11.)

Negatieve marge

Een marge kan negatief zijn. Met een negatieve waarde verschuift een box in de richting van die marge. Dat is het omgekeerde van wat een positieve marge doet, namelijk de box wegdruwen in de tegenovergestelde richting van de marge.

Meestal is er geen reden meer om negatieve marges te gebruiken. Dit werd vooral toegepast om lay-outproblemen op te lossen waarvoor nu nieuwe technieken beschikbaar zijn: flexbox en gridlay-out. Maar u weet nu dat het kan en wat het effect is.

Samengevoegde marges

Om te voorkomen dat tussen boxen dubbele marges ontstaan, worden de marges aan de boven- en onderkant samengevoegd (dat zijn de verticale marges). Wat overblijft is de langste marge van de twee. De Engelse term hiervoor is *collapsing margins*. De marges links en rechts (horizontale marges) worden nooit samengevoegd.

Als vuistregel kunt u aanhouden dat marges worden samengevoegd van aangrenzende elementen in de normale flow. Dus van twee alinea's onder elkaar die een onder- en een bovenmarge hebben, wordt de gemeenschappelijke marge beperkt tot de langste van de twee. Hetzelfde geldt voor twee lijstitems en dergelijke. Ook twee gestapelde div's die beide een boven- en ondermarge hebben, krijgen niet een marge die de optelsom is van die twee, maar een die alleen bestaat uit de grootste waarde.

Een voorbeeld van deze situatie bestaat uit twee gestapelde alinea's `<p>` die (standaard) elk een boven- en ondermarge hebben van 16px. De opgetelde marge tussen de alinea's zou 32px zijn, maar die blijft door samenvoeging 16px. Een ander voorbeeld bestaat uit twee gestapelde div's. De bovenste div heeft een ondermarge van 32px en de onderste div heeft een bovenmarge van 48px. De werkelijke marge is 48px, de langste van de twee.

Een tweede situatie waarin marges worden samengevoegd is als de bovenmarge van de eerste box (een kop of een tekstalinea) in een container grenst aan de bovenmarge van de container. Dat gebeurt als er geen padding en geen rand is. De samengevoegde marge komt aan de buitenkant van de container. In de praktijk ziet u dan dat de kop of tekst tegen de bovenkant van de container staat, wat meestal ook de bedoeling is. Let op: de marge buiten de container kan ook werkelijk invloed hebben op de afstand tot een box die erboven staat. Als die samengevoegde marge breder is dan de marge die de box al had, wordt de ruimte tussen de boxen dus groter! (Zie ook de paragraaf *Block formatting context* in hoofdstuk 10.)

Padding

Padding (opvulling) is de ruimte tussen de inhoud (bijvoorbeeld tekst in een alinea of een afbeelding) en de rand. Als er geen rand is ingesteld, is het vanzelfsprekend de ruimte tot aan de marge.

Voor padding zijn er vier eigenschappen plus een verzameleigenschap:

- padding-top
- padding-right
- padding-bottom
- padding-left
- padding

Padding kan de gebruikelijke waarden hebben, dus pixels, procenten, em, rem enzovoort. Negatieve waarden zijn niet mogelijk. Met de verzameleigenschap padding geeft u waarden op dezelfde manier aan als bij margin. De eerste waarde is voor de bovenkant, de tweede voor de rechterkant, de derde voor de onderkant en de vierde voor de linkerkant. U kunt ook minder dan vier waarden opgeven:

De box van deze tekst is 450px breed en heeft een rand van 5px. Op de tekst is geen padding ingesteld en daarom staat de tekst strak tegen de rands.

Op de tekst in deze box is rondom 20px padding ingesteld. Daardoor blijft de tekst vrij van de rand.

Afbeelding 9.4 Rondom de tekst rechts is 1em (hier 20px) padding ingesteld. Met padding krijgen elementen wat lucht ten opzichte van de rand en omringende elementen (09_04.html).

- padding: 5px de padding is rondom 5 pixels breed;
- padding: 5px 3px boven en onder 5 pixels, rechts en links 3;
- padding: 5px 3px 7px boven 5, rechts 3, onder 7, links 3.

Bij drie waarden wordt de vierde, ontbrekende waarde ontleend aan de tegenovergestelde kant, padding links is dus gelijk aan padding rechts.

Randen

Voor randen (*borders*) zijn meer eigenschappen beschikbaar dan alleen de breedte. In dit hoofdstuk worden de volgende eigenschappen behandeld:

- dikte (breedte)
- kleur
- stijl

In hoofdstuk 12 worden de randeffecten besproken:

- schaduw
- ronde hoeken
- randafbeeldingen

Randdikte

De dikte van de rand kan met de gebruikelijke waarden zoals pixels, procenten, em, rem en zovoort worden aangegeven, plus de waarden `thin`, `medium` (standaard) en `thick`.

Elke randdikte kan afzonderlijk worden ingesteld met:

- `border-top-width`
- `border-right-width`
- `border-bottom-width`
- `border-left-width`
- `border-width`

Net als bij `margin` en `padding` kunnen met de verzameleigenschap `border-width` alle randen ineens worden ingesteld. Bij vier waarden geldt de eerste voor de bovenste rand en de volgende waarden bepalen de randen met de klok mee. Bij minder dan vier waarden geldt hetzelfde als bij `margin` en `padding`.

Randkleur

Voor de kleur van de rand zijn de volgende eigenschappen beschikbaar:

- `border-top-color`
- `border-right-color`
- `border-bottom-color`
- `border-left-color`
- `border-color`

Een randkleur kan een kleurwaarde krijgen (RGB, HSL of kleurnaam), de waarde transparent of het sleutelwoordcurrentColor. De waarde transparent geeft een doorzichtige (onzichtbare) rand die wel gewoon de ingestelde breedte heeft. Met currentColor hebt u in feite een variabele. De kleurwaarde wordt namelijk gehaald uit de waarde van de eigenschap color, dat is de tekstkleur. De rand van de volgende div heeft dezelfde kleur als de tekst.

```
div {  
    color: burlywood;  
    border-color: currentColor;  
    border-width: 8px;  
}
```

Voor het gebruik van de verzameleigenschap border-color gelden dezelfde posities als bij andere verzameleigenschappen voor randen.



Kleuren in CSS

In hoofdstuk 12 worden de kleurwaarden van CSS en de kleursystemen RGB en HSL besproken.

Transparante rand

Bij een transparante (doorschijnende) rand ziet u de achtergrondkleur van het element waarop de rand is ingesteld. De rand hoort immers bij dat element. In de afbeelding heeft de container een kleurig patroon als achtergrond en de alinea heeft een witte achtergrond. We willen in de rand van de alinea de kleurige achtergrond van de container laten doorschijnen. Er is een brede, half-doorzichtige rand; niet transparant (want dat is volledig doorzichtig) maar rgba(255,255,255,.5), dit is halfdoorzichtig wit. Er is nog wel een noodzakelijke extra eigenschap toegevoegd die ervoor zorgt dat de achtergrond stopt aan de buitenkant van de padding (background-clip: padding-box) in plaats van de buitenkant van de rand, het standaardgedrag. In hoofdstuk 12 wordt deze eigenschap uitgelegd.

```
<article>  
    <p>Lorem ipsum dolor sit amet...</p>  
</article>
```

```
article {  
    width: max-content;  
    padding: 1em;  
    background-image: url('../images/patroon-1920.jpg');  
}
```

```

p {
  width: 50ch; /* de eenheid voor lettertekens */
  padding: 2em;
  border: 3px solid rgba(255,255,255,.6);
  margin: 2em;
  background-clip: padding-box;
  background-color: white;
}

```



Afbeelding 9.5 Een halfdoorzichtige witte rand maakt de achtergrondafbeelding van de container zichtbaar (09_05.html).

Analyse van de code

De alinea `<p>` is beperkt tot een breedte van 50ch (dat is vijftig keer de breedte van het teken 0 bij het huidige lettertype). Zonder die beperking zou de alinea net zo breed worden als het browservenster toelaat, want het is een blokbox. Het `<article>` zou met een vaste breedte de `<p>` ook smaller kunnen houden, maar daar is gekozen voor `width: max-content`. Deze instelling zorgt ervoor dat het `<article>` (de container) niet breder wordt dan nodig is voor de tekst plus de bijbehorende padding, rand en marge. Haal deze regel weg en het `<article>` (ook een blokbox) wordt zo breed als zijn container of het browservenster. De achtergrondafbeelding is dan ook over de hele breedte zichtbaar, wat hier zeker niet de bedoeling is. Met een lengtewaarde in plaats van `max-content` zou de breedte van `<article>` moeten worden berekend.

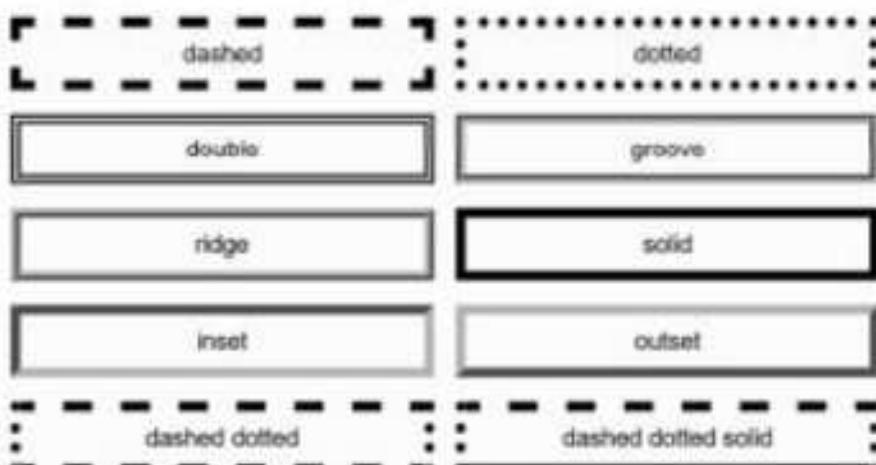
Randstijl

De randstijl wordt ingesteld met de volgende eigenschappen:

- `border-top-style`
- `border-right-style`
- `border-bottom-style`
- `border-left-style`
- `border-style`

Voor de stijl van de rand zijn in nogal wat mogelijkheden beschikbaar:

- none (standaard, geen rand, breedte wordt 0)
- solid (ononderbroken)
- dashed (streeppjes)
- dotted (puntjes)
- double (dubbele rand)
- groove (gekerfd)
- ridge (opstaand)
- inset (ingelegd)
- outset (opgelegd)
- hidden (hetzelfde als none).



Afbeelding 9.6 Voorbeelden van randstijlen (09_06.html).

De randstijl is onmisbaar voor de weergave van randen. De standaardwaarde is none. Bij deze waarde is de breedte automatisch nul, ongeacht wat wordt ingesteld bij border-width. Met andere woorden: als er geen stijl wordt ingesteld, is er geen rand.

Voor het gebruik van de verzameleigenschap border-style gelden dezelfde posities als bij andere verzameleigenschappen.

Korte notatie voor randen

Van elke afzonderlijke zijde van een blok kunnen in één keer de eigenschappen dikte, kleur en stijl worden ingesteld met:

- border-top
- border-right
- border-bottom
- border-left

Geef achtereenvolgens de dikte, de stijl en de kleur aan. De regel voor een dikke, gestreepte, zwarte rand onder een alinea ziet er zo uit:

```
p {
    border-bottom: thick dashed black;
}
```

Ten slotte is er nog de eigenschap border, waarmee in één keer alle randen worden voorzien van een dikte, stijl en kleur. Deze wordt op dezelfde manier gebruikt als de eigenschappen voor de afzonderlijke randen.

```
div {
    border: 1px solid black;
}
```

Waarden die u weglaat worden gereset naar de beginwaarden (medium, none en de kleur van de ouder). Ook border-image (zie hoofdstuk 12) wordt hiermee gereset naar none. Dat betekent dat een declaratie met border voor border-image moet komen.

Overlopende inhoud: overflow

Standaard wordt de box van een element zo groot als nodig is om de inhoud te tonen. Maar het kan zijn dat die box of zijn container een hoogte- of breedtemaat heeft, waardoor die te klein is voor de inhoud. Als er niet voldoende ruimte is voor de inhoud, loopt de box over. Dat wordt *overloop* of *overflow* genoemd.

Lorem ipsum dolor sit amet, consectetur adipisicing elit.
Aut repellat incidente noe, rerum fuga deserunt doloremque
obcaecati qui eaque quaeat consequuntur maiores dolore
deleniti laborum provident dicta ducimus possimus
voluptatem quibusdam. Totam ex non expedita temporibus

sed recusandae, facilis etiam expiscabo animamadu dolores
quibusdam saepe quis perspicatis incidunt a illum molitus
quis harum qui necessitatibus doloremque aliquid dolorum
supsidate? Nihil optio quam provident ullam, ut
implicibus poero illo aut incidunt vitas saepe blanditiis,
excellitas itaque.

Afbeelding 9.7 Er is meer tekst dan er in de box past. In CSS heet dat *overflow*. De doorgetrokken streep markeert de box. De alinea heeft een stippenlijn. De overlopende tekst is halftransparant (*09_07.html*).

De afbeelding toont een container met een rand en een hoogte van 150px. Daarin staat tekst die duidelijk langer is dan erin past. De grijze tekst is de overflow. Wat er met overlopende inhoud gebeurt, wordt bepaald door de eigenschap `overflow`.

De eigenschap `overflow` kan de volgende waarden hebben:

- `visible` (standaard)
- `hidden`
- `clip`
- `scroll`
- `auto`

Met `visible` wordt overlopende inhoud getoond; deze loopt door buiten de box. Het gevolg van `hidden` is dat de inhoud wordt afgekapt en niet wordt getoond; er verschijnt geen schuifbalk en slepen kan ook niet, maar er kan programmatisch een scrollmechanisme worden gemaakt. Met `clip` wordt de overlopende inhoud ook niet getoond en er kan op een enkele manier een scrollmechanisme worden gemaakt. Bij de waarde `scroll` komt er altijd een schuifbalk, ook als de inhoud niet overloopt. Overlopende inhoud wordt weliswaar afgekapt, maar kan met de schuifbalk in beeld worden gebracht. Met `auto` gebeurt vrijwel hetzelfde, met als verschil dat de schuifbalk pas verschijnt als er daadwerkelijk overlopende inhoud is.

Er zijn afzonderlijke eigenschappen voor horizontale en verticale `overflow`: `overflow-x` en `overflow-y`. Deze kunnen dezelfde waarden krijgen als `overflow`.

We hebben hier het voorbeeld gegeven met tekst, maar het principe geldt even goed voor andere inhoud, zoals tabellen of afbeeldingen.



Controleer op overloop

Als tijdens het coderen elementen op een andere plek terechtkomen dan u had verwacht of als er in het browservenster een horizontale schuifbalk verschijnt, controleer dan of er overloop is. De Hulpmiddelen voor ontwikkelaars zijn hiertoe een fantastisch hulpmiddel. Wanneer u een element selecteert in de HTML-weergave, licht dat op in het browservenster en is goed te zien of er overloop is.

De afbeelding toont een container met een rand en een hoogte van 150px. Daarin staat tekst die duidelijk langer is dan erin past. De grijze tekst is de overflow. Wat er met overlopende inhoud gebeurt, wordt bepaald door de eigenschap overflow.

De eigenschap overflow kan de volgende waarden hebben:

- visible (standaard)
- hidden
- clip
- scroll
- auto

Met `visible` wordt overlopende inhoud getoond; deze loopt door buiten de box. Het gevolg van `hidden` is dat de inhoud wordt afgekapt en niet wordt getoond; er verschijnt geen schuifbalk en slepen kan ook niet, maar er kan programmatisch een scrollmechanisme worden gemaakt. Met `clip` wordt de overlopende inhoud ook niet getoond en er kan op een enkele manier een scrollmechanisme worden gemaakt. Bij de waarde `scroll` komt er altijd een schuifbalk, ook als de inhoud niet overloopt. Overlopende inhoud wordt weliswaar afgekapt, maar kan met de schuifbalk in beeld worden gebracht. Met `auto` gebeurt vrijwel hetzelfde, met als verschil dat de schuifbalk pas verschijnt als er daadwerkelijk overlopende inhoud is.

Er zijn afzonderlijke eigenschappen voor horizontale en verticale overflow: `overflow-x` en `overflow-y`. Deze kunnen dezelfde waarden krijgen als `overflow`.

We hebben hier het voorbeeld gegeven met tekst, maar het principe geldt even goed voor andere inhoud, zoals tabellen of afbeeldingen.



Controleer op overloop

Als tijdens het coderen elementen op een andere plek terechtkomen dan u had verwacht of als er in het browservenster een horizontale schuifbalk verschijnt, controleer dan of er overloop is. De Hulpmiddelen voor ontwikkelaars zijn hierbij een fantastisch hulpmiddel. Wanneer u een element selecteert in de HTML-weergave, licht dat op in het browservenster en is goed te zien of er overloop is.

De afbeelding toont een container met een rand en een hoogte van 150px. Daarin staat tekst die duidelijk langer is dan erin past. De grijze tekst is de overflow. Wat er met overlopende inhoud gebeurt, wordt bepaald door de eigenschap overflow.

De eigenschap overflow kan de volgende waarden hebben:

- visible (standaard)
- hidden
- clip
- scroll
- auto

Met `visible` wordt overlopende inhoud getoond; deze loopt door buiten de box. Het gevolg van `hidden` is dat de inhoud wordt afgekapt en niet wordt getoond; er verschijnt geen schuifbalk en slepen kan ook niet, maar er kan programmatisch een scrollmechanisme worden gemaakt. Met `clip` wordt de overlopende inhoud ook niet getoond en er kan op een enkele manier een scrollmechanisme worden gemaakt. Bij de waarde `scroll` komt er altijd een schuifbalk, ook als de inhoud niet overloopt. Overlopende inhoud wordt weliswaar afgekapt, maar kan met de schuifbalk in beeld worden gebracht. Met `auto` gebeurt vrijwel hetzelfde, met als verschil dat de schuifbalk pas verschijnt als er daadwerkelijk overlopende inhoud is.

Er zijn afzonderlijke eigenschappen voor horizontale en verticale overflow: `overflow-x` en `overflow-y`. Deze kunnen dezelfde waarden krijgen als `overflow`.

We hebben hier het voorbeeld gegeven met tekst, maar het principe geldt even goed voor andere inhoud, zoals tabellen of afbeeldingen.



Controleer op overloop

Als tijdens het coderen elementen op een andere plek terechtkomen dan u had verwacht of als er in het browservenster een horizontale schuifbalk verschijnt, controleer dan of er overloop is. De Hulpmiddelen voor ontwikkelaars zijn hierbij een fantastisch hulpmiddel. Wanneer u een element selecteert in de HTML-weergave, licht dat op in het browservenster en is goed te zien of er overloop is.

De toekomst: logische eigenschappen

Alles wat hiervoor is beschreven over breedte, hoogte, marge, padding en randen bestaat al jaren en werkt in alle browsers. Maar de ontwikkelingen staan niet stil. Bij de ontwikkeling van flexbox en gridlay-out is het denken over links, rechts, boven en onder gemoderniseerd. Dat heeft alles te maken met de schrijfrichting, in CSS in te stellen als de eigenschap `writing-mode` (zie ook hoofdstuk 11, *Tekst en typografie*).

Het Nederlands (en Engels, maar ook andere op het Latijn gebaseerde systemen zoals Grieks en Cyrilisch) wordt geschreven van linksboven naar rechtsonder (`writing-mode: horizontal-tb`). Dan is het vanzelfsprekend dat links altijd het begin van de regel is en rechts het eind, net zoals boven altijd het begin van een tekstblok is en onder het eind.

Er zijn ook talen die een andere schrijfrichting hebben. Op Han gebaseerde systemen (waaronder Japans, Chinees en Koreaans, CJK-talen) worden weliswaar steeds vaker net als het Latijnse systeem geschreven, maar gaan traditioneel van boven naar beneden en van rechts naar links; de regel staat dus rechtop (`writing-mode: vertical-rl`).

In het Arabische systeem (Arabisch en Hebreeuws) loopt net als in het Nederlands de regel horizontaal, maar van rechts naar links. Dat wordt overigens niet ingesteld met `writing-mode`, maar met het attribuut `dir` op het element `<html>` (zie hoofdstuk 2):

```
<html lang="ar" dir="rtl">
```

In het Nederlands is het begin van het tekstblok dus inderdaad top en het begin van de regel is altijd left. In het Arabisch is het begin van het tekstblok ook top, maar het begin van de regel is right. In het Japans wordt de verwarring compleet, want het tekstblok begint bij right en de regel begint bij top. Daarom is er een systeem bedacht dat bij elke schrijfrichting vanzelf het echte begin en eind van een blok of regel aanduidt: dit zijn logische waarden relatief ten opzichte van de richting van de inhoud (*CSS Logical Properties and Values Level 1* – nog een concept – staat op www.w3.org/TR/css-logical-1/). De bestaande waarden worden ook wel fysieke waarden genoemd.

Breedte en hoogte

Een breedte wordt altijd gezien als een afmeting die loopt in de richting van de tekstregel, in de CSS-specificaties wordt dat *inline* genoemd. Een hoogte is de maat die loopt in de richting waarin de regels/blokken op elkaar worden gestapeld, die richting wordt *block* genoemd.

Volgens die denkwijze wordt de breedte (*width*) de *inline-size*. De hoogte (*height*) wordt de *block-size*. De volgende CSS maakt een blok van 100px hoog en 300px breed:

```
.box {
  block-size: 100px;
  inline-size: 300px;
}
```

Het maakt nu niet meer uit welke taal en tekstrichting er worden gebruikt. De *inline-size* en de *block-size* geven altijd de breedte en de hoogte volgens het ingestelde taalsysteem.

Het einde van top, bottom, left, right?

Eigenschappen die de sleutelwoorden *top*, *bottom*, *left* en *right* accepteren, kunnen in plaats daarvan nieuwe sleutelwoorden gebruiken. Het is niet verplicht om ze te gebruiken en de oude waarden blijven geldig. Het heeft wel voordelen om ermee te leren werken (zie de volgende paragraaf). In de tabel is te zien welke nieuwe waarden horen bij de bestaande waarden op een Nederlandse website (Latijns systeem).

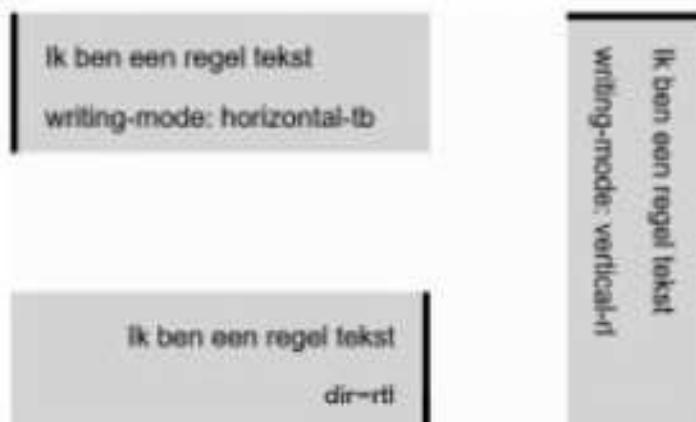
Huidig	Nieuw
<i>top</i>	<i>*-block-start</i>
<i>bottom</i>	<i>*-block-end</i>
<i>left</i>	<i>*-inline-start</i>
<i>right</i>	<i>*-inline-end</i>

Deze nieuwe waarden worden gebruikt voor marge, padding en randen. Het sterretje wordt dan vervangen door margin, padding of border. Een paar voorbeelden gebaseerd op de Latijnse schrijfrichting:

- *margin-left* wordt *margin-inline-start*
- *padding-bottom* wordt *padding-block-end*
- *border-right* wordt *border-inline-end*

Bij border kunnen de eigenschappen voor breedte, kleur en stijl worden toegevoegd:

- border-top-width wordt border-block-start-width
- border-right-color wordt border-inline-end-color
- border-bottom-style wordt border-block-end-style



Afbeelding 9.8 Verschillende instellingen voor `writing-mode`. Let op de zwarte rand, die staat vanzelf altijd aan het begin van de regel met `border-inline-start` (09_08.html).

Hetzelfde systeem kan ook worden gebruikt bij onderwerpen die verderop in dit hoofdstuk worden besproken: positionering, float en clear, en resize. Het is ook van toepassing op de uitlijning van bijschriften bij tabellen (`caption-side`), de uitlijning van tekst (`text-align`, hoofdstuk 11) en afgeronde hoeken (`border-radius`, hoofdstuk 12).

Wat is het nut?

Waarom zou u (ook) leren werken met de waarden voor block en inline in plaats van top, bottom, left en right? Daar zijn een paar redenen voor.

- Als dit uw eerste kennismaking met CSS is, leert u van het begin af aan denken in een toekomstbestendig systeem.
- Wie weet of u ooit te maken krijgt met webpagina's met tekst die in het Arabische of Han-systeem wordt geschreven? De wereld is een dorp, nietwaar?
- Met verschillende instellingen voor de schrijfrichting (`writing-mode`) kunnen grafische effecten worden gemaakt (zie hoofdstuk 11). Ook daarbij bieden de nieuwe waarden praktisch voordeel.
- Nieuwe lay-outmethoden, flexbox en gridlay-out, werken al met de begrippen start en end. U zult ze beter begrijpen wanneer vertrouwd bent met de logische waarden.

Praktische beperking

Zoals zo vaak bepaalt de browserondersteuning of CSS-eigenschappen praktisch bruikbaar zijn en daar zit op het moment van schrijven wel een probleempje. In de eerste helft van 2019 zijn in Chrome en Firefox (desktop en mobiel) de hier genoemde eigenschappen bruikbaar. Voor Safari, iOS Safari en Samsung Internet is de prefix `-webkit-` nodig. Edge weet nog nergens van. Voor productie is het daarom nodig terugvalwaarden te gebruiken of om voorlopig alleen de huidige 'onlogische' waarden te gebruiken.



Hoe wordt een terugvalwaarde gebruikt?

De browser negeert eigenschappen die niet worden herkend. Dat maakt het mogelijk om bijvoorbeeld eerst een `width` in te stellen; die eigenschap snap de browsers zeker. Schrijf in de volgende regel een `inline-start`. Wordt die niet herkend, dan gebruikt de browsers gewoon de `width`.

De toekomst: box alignment

Het uitlijnen van de inhoud van blokken is altijd nogal lastig geweest, met als grootste struikelblok verticaal uitlijnen in een container. Met de komst van flexbox en gridlay-out zijn nieuwe eigenschappen geïntroduceerd waarmee de uitlijning in alle richtingen veel eenvoudiger is geworden. Buiten deze lay-outsysteem werken die eigenschappen alleen nog niet, maar dat komt eraan. De conceptversie van de module *CSS Box Alignment level 3* is te vinden op www.w3.org/TR/css-align-3/. Daarin staat een reeks eigenschappen voor het uitlijnen en verdelen van inhoud:

- `justify-content`
- `align-content`
- `place-content`
- `justify-items`
- `align-items`
- `place-items`
- `justify-self`
- `align-self`
- `place-self`
- `row-gap`
- `column-gap`
- `gap`

De eigenschappen met `content`, `items` en `self` kunnen in verschillende lay-outsysteem verschillende effecten hebben. Dit wordt duidelijk bij de uitleg

van flexbox en gridlay-out in hoofdstuk 10. Toch is de kern hetzelfde: justify gaat altijd over uitlijnen in de regerichting en align lijnt uit in de blokrichting. Zoals beschreven in de paragraaf *De toekomst: logische waarden* is op een Nederlandse website de richting van de regel van links naar rechts en blok (de stapelvolgorde van de regels) van boven naar beneden. De drie eigenschappen met place-* maken het mogelijk om justify en align in één declaratie te plaatsen (een verzamelwaarde vergelijkbaar met bijvoorbeeld margin, waar mee alle marges van een box tegelijk worden ingesteld). De drie gap-eigen schappen regelen de tussenruimte tussen boxen.

Deze korte uitleg is niet het complete verhaal, maar bedoeld als voorbereiding op een andere manier van denken over uitlijnen. De onderdelen die al bruikbaar zijn, komen aan bod bij flexbox en gridlay-out.

Weergavemodel

Hiervoor hebben we het gehad over het formaat en de uiterlijke eigenschappen van individuele boxen. Een pagina bestaat echter uit een (grote) verzameling van boxen die op de een of andere manier op elkaar reageren. Het resultaat daarvan is de lay-out.

Alinea's komen standaard onder elkaar te staan, net als items in een lijst. Maar bijvoorbeeld hyperlinks en afbeeldingen worden naast elkaar gezet. Tenminste, tot de 'regel' vol is, want dan gaan ze op de volgende 'regel' verder. Blijkbaar is er een mechanisme dat HTML-elementen een bepaalde standaardweergave geeft en dat verschil maakt tussen 'dingen naast elkaar zetten' en 'dingen onder elkaar zetten'. Verantwoordelijk daarvoor is het weergavemodel.

Het weergavemodel is een Nederlandse vertaling van het *visual formatting model*, zoals het mechanisme wordt genoemd. Het hiervoor besproken boxmodel, waarbij de afmetingen van de boxen worden bepaald, is de eerste stap in het weergavemodel. De volgende stap is het verdelen van de boxen op de pagina: de lay-out. Het weergavemodel geeft elementen eigenschappen die invloed hebben op hun eigen weergave en de relatie met andere elementen. Zo nemen bepaalde elementen de hele breedte van een venster in beslag, zelfs als dat voor het weergeven van de inhoud niet nodig is (of preciezer gezegd: de hele breedte van hun container). Andere nemen alleen de ruimte die nodig is om de inhoud van het element weer te geven en laten de ruimteernaast vrij voor andere elementen.

Achter de schermen

Dat 'dingen naast elkaar' en 'dingen onder elkaar' komen te staan, wordt in CSS bepaald door het type box. Om dat goed te kunnen begrijpen moeten we kijken naar het ontstaan van de boxen.

Om een HTML-pagina te kunnen weergeven, wordt van het bestand een boomstructuur gemaakt van elementen en tekst (zie ook *De boomstructuur* in hoofdstuk 8). Voordat de elementen daadwerkelijk worden gerenderd (getekend in het browsersvenster), wordt een tussenstap gemaakt waarbij alle CSS-eigenschappen van alle elementen een waarde krijgen. (Zie ook *CSS-verwerking door de browser* in hoofdstuk 8). Het resultaat hiervan is een boomstructuur van nul of meer boxen per element. Bij de meeste elementen wordt één box gemaakt, maar als in de CSS de weergave van dat element is uitgeschakeld, wordt er geen box gemaakt. Voor een element zoals een lijstitem worden twee boxen gemaakt: een buitenste box voor het lijstitem zelf en daarbinnen een tweede box voor het opsommingsteken. De box is dus een container of doos voor het element en zijn inhoud. Het type box en daarmee het effect op de lay-out is afhankelijk van de waarde van de eigenschap `display`.

De eigenschap `display`

Ieder HTML-element heeft een standaardinstelling voor `display`, gebaseerd op de beschrijving/functie in de HTML-specificatie. Daarom begint elk element `<p>` op een nieuwe regel en gebruikt het alle beschikbare breedte en loopt met `` benadrukte tekst gewoon door in de tekstregel. De waarde van `display` kan echter worden veranderd om bijvoorbeeld een lay-out te maken met flexbox of gridlay-out.

Sinds de introductie van CSS3 heeft `display` een eigen module, *CSS Display level 3* (www.w3.org/TR/css-display-3/). Deze is sinds augustus 2018 uit de conceptfase, maar dat betekent niet per se dat alles blijft zoals het is. Desondanks is in de praktijk niet te verwachten dat er nog grote veranderingen komen.

Waarden voor binnen en buiten

Zoals gezegd, de waarde van `display` bepaalt hoe een box zich gedraagt. In technische termen: de eigenschap `display` definieert het weergavetype, dat bestaat uit twee kenmerken:

- het gedrag van de box ten opzichte van andere boxen, dit wordt het *outer display type* genoemd;
- het gedrag van de afstammelingen van de box, dit is het *inner display type*.

Hoewel het de bedoeling is dat de weergavetypen van `display` afzonderlijk kunnen worden ingesteld, is dat op dit moment niet mogelijk. Gebruik daarom in de CSS-code voor `display` altijd één waarde, net als voorheen met CSS 2. Wanneer de ene waarde van `display` wordt ingesteld, is de tweede waarde daar automatisch aan gekoppeld. In CSS 2 bestonden overigens al waarden die de twee typen verbond met een koppelteken en die staan hierna onder *Gekoppelde waarden*.

Buitenste weergavetypen

Bij deze groep gaat het om de samenwerking tussen de box en de omringende boxen in de normale flow.

- `block` Maakt een blokcontainer (blokbox). Dat is een box die wordt gestapeld.
- `inline` Maakt een box die in de (tekst)regel wordt geplaatst.
- `run-in` Maakt een inlopende box, wat betekent dat die probeert zichzelf te plaatsen in de volgende blokbox, aan het begin van de regelinhoud van die box. Deze waarde heeft ooit wel gewerkt, maar wordt door geen enkele browser meer ondersteund (2019).

De inlopende kop Deze alinea is een `block-level box` en de cursieve tekst in de (rode) rechthoek is een inline-level box. De kop heeft `display: run-in`, maar de weergave is een simulatie, want de browser is dit trucje verleerd.

Afbeelding 9.9 Voorbeelden van de waarden die horen bij `display: outside`. De alinea is `block`, de cursieve tekst is `inline` en de kop is `run-in` (simulatie) ([09_09.html](#)).

Binnenste weergavetypen

Bij deze groep gaat het erom hoe de inhoud van de box wordt gelay-out.

- `flow` Dit is de normale plaatsing van boxen. Elk elementen doet wat zijn `display`-waarde zegt. Wanneer de `display`-waarde van een element `block`, `inline`, `run-in` of `list-item` is, is het binnenste weergavetype automatisch `flow`.
- `flow-root` Net als `flow`, maar met als belangrijk verschil dat een nieuwe `block formatting context` wordt gemaakt. Een `block formatting context` (BFC) is een soort privéterrein binnen de normale flow, met een hek om de inhoud. Een element dat een `block formatting context` maakt, omvat alle elementen die erin zitten; het voorkomt dat floats buiten de container kunnen staan, het voorkomt dat tekst om floats loopt en het voorkomt het samenvoegen van marges. Floats worden in hoofdstuk 10 besproken.
- `table` Maakt een blokbox voor een tabel en maakt een nieuwe `block formatting context`.

- **flex** Maakt blokbox waarin de kindelementen flexitems zijn (*Flexible Box Layout*). Het maakt een *flex formatting context*.
- **grid** Maakt een blokbox waarin de kindelementen griditems zijn (*Grid Layout*). Het maakt een *grid formatting context*.



Block/flex/grid formatting context

De formatting context bepaalt hoe de elementen zich binnen die context gedragen. Een element in een flex formatting context heeft andere eigenschappen dan een element in een grid formatting context. Praktisch gezien is *formatting context* de container/het omvattende blok. De specifieke eigenschappen gelden alleen binnen die container.

Lijstitems

- **list-item** Maakt een blokbox. Heeft als speciale eigenschap dat een pseudo-element ::marker-box wordt gemaakt. Daarin komt het opsommingsteken dat is ingesteld in de eigenschappen voor *list-style*.

Tabelwaarden van display

Een tabel heeft een complexe interne structuur en speciaal daarvoor zijn de interne displaywaarden `table-row-group`, `table-header-group`, `table-footer-group`, `table-row`, `table-cell`, `table-column-group`, `table-column` en `table-caption`. Deze waarden zult u (bijna) nooit gebruiken. Een tabel die goed is opgezet met HTML heeft automatisch de juiste display-waarden.

De waarden `none` en `contents`

- **none** Het element en zijn afstammelingen maken geen boxen en worden daardoor niet weergegeven. Dit kunt u zien met de Hulpmiddelen voor ontwikkelaars. Selecteer het element in de code en zie dat in de paginaweergave geen pop-up met afmetingen verschijnt.
Let op, ook in spraakbrowsers wordt inhoud met `display: none` niet voorgelezen. Daardoor kan voor slechtzienden belangrijke informatie verloren gaan en daarom is dit geen goede manier om elementen alleen 'voor het oog' te verbergen. Met de eigenschap `visibility` kan de zichtbaarheid van elementen op andere manieren worden beïnvloed. Zie verderop in dit hoofdstuk.
- **contents** Dit is een nieuwe waarde met een bijzondere (handige!) eigenschap. Een element met `display: contents` maakt zelf geen box, maar de kinderen en pseudo-elementen wel en de tekstloop is normaal. Een typische toepassing hiervoor is een lijst in `gridlay-out`. Een kenmerk van `gridlay-out` (en `flexbox`) is dat deze lay-outtypen alleen invloed hebben op de kindelementen, niet op de kleinkinderen enzovoort. Lijstitems `` die per definitie in een `` of `` staan, kunnen daardoor niet zomaar

- **flex** Maakt blokbox waarin de kindelementen flexitems zijn (*Flexible Box Layout*). Het maakt een *flex formatting context*.
- **grid** Maakt een blokbox waarin de kindelementen griditems zijn (*Grid Layout*). Het maakt een *grid formatting context*.



Block/flex/grid formatting context

De formatting context bepaalt hoe de elementen zich binnen die context gedragen. Een element in een flex formatting context heeft andere eigenschappen dan een element in een grid formatting context. Praktisch gezien is *formatting context* de container/het omvattende blok. De specifieke eigenschappen gelden alleen binnen die container.

Lijstitems

- **list-item** Maakt een blokbox. Heeft als speciale eigenschap dat een pseudo-element `::marker-box` wordt gemaakt. Daarin komt het opsommingsteken dat is ingesteld in de eigenschappen voor `list-style`.

Tabelwaarden van display

Een tabel heeft een complexe interne structuur en speciaal daarvoor zijn de interne displaywaarden `table-row-group`, `table-header-group`, `table-footer-group`, `table-row`, `table-cell`, `table-column-group`, `table-column` en `table-caption`. Deze waarden zult u (bijna) nooit gebruiken. Een tabel die goed is opgezet met HTML heeft automatisch de juiste displaywaarden.

De waarden `none` en `contents`

- **none** Het element en zijn afstammelingen maken geen boxen en worden daardoor niet weergegeven. Dit kunt u zien met de Hulpmiddelen voor ontwikkelaars. Selecteer het element in de code en zie dat in de paginaweergave geen pop-up met afmetingen verschijnt.
Let op, ook in spraakbrowsers wordt inhoud met `display: none` niet voorgelezen. Daardoor kan voor slechtzienden belangrijke informatie verloren gaan en daarom is dit geen goede manier om elementen alleen 'voor het oog' te verbergen. Met de eigenschap `visibility` kan de zichtbaarheid van elementen op andere manieren worden beïnvloed. Zie verderop in dit hoofdstuk.
- **contents** Dit is een nieuwe waarde met een bijzondere (handige!) eigenschap. Een element met `display: contents` maakt zelf geen box, maar de kinderen en pseudo-elementen wel en de tekstloop is normaal. Een typische toepassing hiervoor is een lijst in gridlay-out. Een kenmerk van gridlay-out (en flexbox) is dat deze lay-outtypen alleen invloed hebben op de kindelementen, niet op de kleinkinderen enzovoort. Lijstitems `` die per definitie in een `` of `` staan, kunnen daardoor niet zomaar



Bekijk waarden van display

Het is nuttig om in de Hulpmiddelen voor ontwikkelaars van een HTML-pagina de waarden van `display` bij verschillende elementen te bekijken. U zult dan zien dat een `<p>` de waarde `block` heeft, `` de waarde `inline`, `` de waarde `block` en `` de waarde `list-item`, om maar wat voorbeelden te noemen.

Inhoud (niet) weergeven: visibility

Eerder kwam de eigenschap `display: none` voorbij, waarmee een element niet wordt weergegeven en er zelfs geen box wordt gemaakt. In de lay-out lijkt het alsof het nooit heeft bestaan. Met `visibility` kan een element ook worden verborgen, maar er wordt wel een box gemaakt en de ruimte blijft voor dat onzichtbare element gereserveerd. De bruikbare waarden zijn:

- `visible` het element is zichtbaar (standaard);
- `hidden` het element wordt verborgen.

Het gebruik van `visibility` heeft wel een belangrijk nadeel: visueel verborgen inhoud is ook niet beschikbaar voor screenreaders. Daarmee kunt u een site nutteloos maken voor blinde en slechtziende bezoekers.

Niet verbergen voor screenreaders

In grafische browsers is sommige (tekst)informatie niet nodig voor de ziende gebruiker, maar is die voor iemand die een screenreader gebruikt juist onmisbaar. Een bekend voorbeeld is `skip-links`, een koppeling waarmee de gebruiker van een spraakbrowsers de navigatie kan overslaan om direct naar de hoofdinhoud te gaan. Dergelijke inhoud kan niet worden verborgen met `visibility: hidden` of `display: none`. In beide gevallen kan een element niet meer de focus krijgen (het is onbereikbaar voor de Tab-toets) en daarmee is het element onbruikbaar geworden.

De oplossing is een klasse die vaak `visually-hidden` of vergelijkbaar wordt genoemd. Deze maakt het element onzichtbaar in een grafische browser en zorgt er tegelijkertijd voor dat een screenreader de inhoud wel kan gebruiken. De volgende CSS-declaratie is een aanbeveling van WebAIM, een organisatie die oplossingen voor webtoegankelijkheid bedenkt voor mensen met beperkingen (webaim.org).

```
.hidden {
    position: absolute;
```

```
left:-10000px;
top:auto;
width:1px;
height:1px;
overflow:hidden;
}
```

Analyse van de code

Het element wordt absoluut geïnitialiseerd (zie hoofdstuk 10) en de hoek linksboven wordt tienduizend pixels naar links verplaatst. Daarmee staat het ver buiten het zichtbare gebied. De regel `top: auto` is om te voorkomen dat in sommige situaties de verplaatsing naar links mislukt. De rest is ook voor de zekerheid. Het element krijgt een formaat van 1 x 1 pixels en wat daar niet in past wordt verborgen (`overflow: hidden`). Als de positionering mislukt, is daardoor het element in ieder geval niet te zien.

Samenvatting

In dit hoofdstuk is uitgelegd wat er achter de schermen nodig is om van HTML-elementen een opgemaakte pagina te maken. Deze kennis is onmisbaar bij het maken van lay-outs.

- Alles begint met het boxmodel. Van elk HTML-element wordt een box gemaakt en elke box heeft dezelfde eigenschappen: `width`, `height`, `margin`, `padding` en `border`.
- `width` en `height` zijn de breedte en de hoogte van een box. Deze kunnen worden ingesteld op een vaste waarde zoals pixels of een percentage, op een waarde afgeleid van de inhoud, zoals `min-content` of `max-content`, of met de standaardwaarde `auto` een formaat krijgen dat past bij de inhoud en de beschikbare ruimte.
- `margin` is de ruimte aan de buitenkant van een box. Een marge is altijd transparant.
- `border` is de rand tussen de marge en de padding. Een rand heeft een dikte, een stijl en een kleur. Een rand zonder stijl is altijd 0.
- `padding` is de ruimte tussen de inhoud van de box en de rand.
- De eigenschap `box-sizing` bepaalt of de ingestelde hoogte/breedte geldt voor de inhoud (dan komen padding en border er nog bij en wordt de totale box groter) of dat die inclusief inhoud, padding en rand is (de box blijft even groot, maar de inhoud krijgt minder ruimte).
- Als een box te klein is voor de inhoud, loopt de inhoud over. Dit heet `overflow`. Standaard is overlopende inhoud zichtbaar buiten de box, maar inhoud kan ook binnen de box worden gehouden en dan verschijnen er schuifbalken.

- Er wordt gewerkt aan een CSS-module die nieuwe eigenschappen definieert voor `width`, `height`, `top`, `bottom`, `left` en `right`. In de module *CSS Logical Properties and Values level 1* worden de nieuwe eigenschappen gebaseerd op de schrijfrichting. De breedte loopt altijd in de richting van de regel (`inline`) en de hoogte in de stapelrichting van de regels (`block`). Op een Nederlandse pagina wordt `left` dan `inline-start` en `bottom` wordt `block-end`.
- In een andere nieuwe module, *CSS Box Alignment level 3*, worden de uitlijning en verdeling van boxen gedefinieerd. Deze module bevat eigenschappen zoals `justify-content` en `align-items` die nu al worden gebruikt bij `gridlay-out` en `flexbox`, maar later bij alle boxen bruikbaar moeten zijn.
- Na het maken van de boxen volgens het boxmodel wordt het gedrag van de boxen bepaald door het visual formatting model. Dit zorgt ervoor dat ‘dingen naast elkaar’ en ‘dingen onder elkaar’ komen te staan. Dat wordt bepaald door het type box en dat wordt vastgelegd in de eigenschap `display`.
- Elk HTML-element heeft standaard een waarde voor `display` die is gebaseerd op de betekenis en functie van het elementen. Daarom staan alinea’s onder elkaar (`block`) en worden hyperlinks en afbeeldingen naast elkaar gezet (`inline`). Met CSS kan de waarde van `display` worden veranderd, waarmee lay-outmethoden zoals `gridlay-out` en `flexbox` worden ingeschakeld.

Het volgende hoofdstuk gaat over het daadwerkelijk maken van lay-outs, te beginnen bij het positioneren van inhoud en floats tot complete pagina-indelingen met `gridlay-out`.

Oefeningen

- Maak een pagina met een aantal tekstblokken (willekeurige alineatekst is voldoende, een regel of vijf is genoeg; typ in Visual Studio Code de opdracht `p>lorem`) en plaats die blokken in afzonderlijke containers, bijvoorbeeld zo:

```
<div>
  <p> ... </p>
</div>
```

- Maak een stijlregel met een elementselector om de blokken verschillende instellingen te geven voor:
 - het formaat (breedte en hoogte – laat die eigenschappen ook een keer weg om te zien wat het effect is);
 - de padding en de marge;

- de rand – speel met dikte, kleur, randstijl en verschillende instellingen op de zijden;
- met de eigenschap `background-color` kunt u de blokken herkenbaarder maken – gebruik grijswaarden zoals #666 of #aaa, of zoek in hoofdstuk 12 op hoe u elke andere kleur kunt gebruiken.
- Kijk wat er gebeurt wanneer u de `box-sizing` aanpast. Probeer twee of meer blokken in een container te plaatsen, zoals in dit hoofdstuk is beschreven.



Gebruik de voorbeeldcode

Van vrijwel alle afbeeldingen is de HTML- en CSS-code beschikbaar. De bestandsnaam staat in het bijschrift. Gebruik deze code om te oefenen: pas de code aan, voeg dingen toe, speel ermee! Kijk op handboek-html-css.nl.

Lay-outs maken met CSS

Met de kennis over het boxmodel en het weergavemodel uit het vorige hoofdstuk kan eindelijk worden begonnen aan het leuke werk: het maken van lay-outs. CSS biedt daar geavanceerde mogelijkheden voor, met gridlay-out en flexbox als laatste toevoegingen. In dit hoofdstuk worden alle mogelijkheden besproken, met voorbeelden van bekende Nederlandse websites. Er is echter een probleem: u weet niet op wat voor scherm de webpagina wordt bekeken. De oplossing is: laat de pagina zich automatisch aanpassen aan het schermformaat. Dit heet responsive design en media queries zijn daar een hulpmiddel voor.

U leert in dit hoofdstuk:

Flexibele Pagina's bouwen met responsive design en media queries.

Hoe CSS-positionering werkt: normale flow, static, relatief, absoluut, fixed en sticky.

De functie van floats in moderne lay-outs.

Wanneer multi-column een goede optie is en hoe dat werkt.

Complete pagina-indelingen maken met gridlay-out.

Flexibele componenten bouwen met flexbox.

Hoofdstuk
10

Inleiding

Na het maken van boxen volgens het boxmodel en de eerste weergave van die boxen volgens het weergavemodel en de eigenschap `display`, wat wordt gerekend door de browser, wordt het tijd om zelf in actie te komen. *It's layout time!* Geen enkele lay-out is gebouwd met maar één lay-outsysteem. Het is altijd een combinatie van de volgende mogelijkheden:

- normale flow
- positionering (relatief, absoluut, fixed)
- float
- meerdere tekstkolommen
- gridlay-out
- flexbox

Het startpunt is altijd de normale flow. Zeker op de kleine schermen van mobiele apparaten vallen veel onderdelen vanzelf op hun plek. Pas als het scherm groter wordt, wordt het nodig om componenten op hun plek te zetten. Hiermee hebben we gelijk een ander kernpunt van CSS-lay-outs te pakken: flexibel design of beter: *responsive design*. In een wereld waarin steeds meer mensen websites bekijken op mobiele apparaten met een enorme variatie aan schermformaten, is het onmogelijk om een website op één formaat te bouwen. Vandaar dat we beginnen met een crash course responsive design.

Responsive design

Een webpagina moet op elk schermformaat bruikbaar zijn. Hij hoeft er niet altijd hetzelfde uit te zien (dat kan ook niet), maar de informatie moet in het scherm passen en leesbaar zijn, knoppen en links moeten groot genoeg zijn om erop te kunnen tikken en formulieren moeten bruikbaar zijn, om maar een paar dingen te noemen. Omdat een webontwikkelaar niet weet of de bezoeker een smartphone, een tablet, een laptop of een bureaucomputer gebruikt, moeten er verschillende versies van het ontwerp worden gemaakt. Niet precies op maat voor een bepaalde schermgrootte, maar met overgangen naar een andere lay-out op punten waar het ontwerp niet meer klopt ('breekt'). Om praktische redenen wordt de CSS vaak opgebouwd vanaf het kleinste scherm. Daarna kan met de responsive-design-modus van de Hulpmiddelen voor ontwikkelaars in de browser het schermformaat geleidelijk groter worden gemaakt. Zodra het ontwerp breekt, wordt de lay-out aangepast. Deze aanpak heet *mobile first*. Een andere aanpak is gebaseerd op een doorsnee van apparaatafmetingen. Het UI-framework Bootstrap gebruikt bijvoorbeeld de verdeling < 576px, ≥ 576px, ≥ 768px, ≥ 992px, ≥ 1200px.

Inleiding

Na het maken van boxen volgens het boxmodel en de eerste weergave van die boxen volgens het weergavemodel en de eigenschap `display`, wat wordt gerekend door de browser, wordt het tijd om zelf in actie te komen. *It's layout time!* Geen enkele lay-out is gebouwd met maar één lay-outsysteem. Het is altijd een combinatie van de volgende mogelijkheden:

- normale flow
- positionering (relatief, absoluut, fixed)
- float
- meerdere tekstkolommen
- gridlay-out
- flexbox

Het startpunt is altijd de normale flow. Zeker op de kleine schermen van mobiele apparaten vallen veel onderdelen vanzelf op hun plek. Pas als het scherm groter wordt, wordt het nodig om componenten op hun plek te zetten. Hiermee hebben we gelijk een ander kernpunt van CSS-lay-outs te pakken: flexibel design of beter: *responsive design*. In een wereld waarin steeds meer mensen websites bekijken op mobiele apparaten met een enorme variatie aan schermformaten, is het onmogelijk om een website op één formaat te bouwen. Vandaar dat we beginnen met een crash course responsive design.

Responsive design

Een webpagina moet op elk schermformaat bruikbaar zijn. Hij hoeft er niet altijd hetzelfde uit te zien (dat kan ook niet), maar de informatie moet in het scherm passen en leesbaar zijn, knoppen en links moeten groot genoeg zijn om erop te kunnen tikken en formulieren moeten bruikbaar zijn, om maar een paar dingen te noemen. Omdat een webontwikkelaar niet weet of de bezoeker een smartphone, een tablet, een laptop of een bureaucomputer gebruikt, moeten er verschillende versies van het ontwerp worden gemaakt. Niet precies op maat voor een bepaalde schermgrootte, maar met overgangen naar een andere lay-out op punten waar het ontwerp niet meer klopt ('breekt'). Om praktische redenen wordt de CSS vaak opgebouwd vanaf het kleinste scherm. Daarna kan met de responsive-design-modus van de Hulpmiddelen voor ontwikkelaars in de browser het schermformaat geleidelijk groter worden gemaakt. Zodra het ontwerp breekt, wordt de lay-out aangepast. Deze aanpak heet *mobile first*. Een andere aanpak is gebaseerd op een doorsnee van apparaatafmetingen. Het UI-framework Bootstrap gebruikt bijvoorbeeld de verdeling < 576px, ≥ 576px, ≥ 768px, ≥ 992px, ≥ 1200px.

Maar hoe weet de browser wanneer de lay-out moet veranderen? Soms gaat dat bijna vanzelf goed (`gridlayout` is er behoorlijk goed in), maar meestal zijn daarvoor media queries nodig.



Afbeelding 10.1 Media queries in actie: `belastingdienst.nl` op drie formaten.

Media queries

Media queries zijn een mechanisme om opmaakkenmerken te koppelen aan eigenschappen van een weergavemedium. Het bestaat al sinds HTML 4 en CSS 2.1, maar dat was niet meer dan een begin. In de CSS3-module *Media Queries* uit 2012 (www.w3.org/TR/css3-mediaqueries/) zijn de mogelijkheden uitgebreid en verfijnd en sinds 2017 is er de module *Media Queries level 4* (www.w3.org/TR/mediaqueries-4/) met weer nieuwe aanpassingen. Er kan rekening worden gehouden met de venstergrootte en er zijn meer criteria, zoals de resolutie, de beeldverhouding en de oriëntatie (portret of landschap).

Media queries bestaan uit een apparaattype en mediakenmerken (*media features*), zoals een vensterbreedte. Als het apparaattype overeenkomt en ook de kenmerken kloppen, worden de bijbehorende stijlregels uitgevoerd. Een media

query heeft de volgende algemene vorm (let op het sleutelwoord and en de haakjes om de expressie):

```
@media apparaat and (media feature) {  
    selector {  
        declaraties;  
    }  
}
```

- Elke media query begint met @media.
- Waarden voor *apparaat* zijn screen (beeldscherm), print (printer) en speech (spraak). Als u niets instelt is de waarde is all.
- Waarden voor *media feature* zijn bijvoorbeeld min-width of orientation (zie hierna).
- Het sleutelwoord and wordt gebruikt om expressies te combineren.
- Alle inhoud van de media query staat tussen accolades.
- Elke stijlregel heeft de gebruikelijke accolades.
- Media queries voor verschillende apparaten worden gescheiden door een komma. In feite is de komma een of-operator:

```
@media screen and (feature), print and (feature) /* of screen en feature, of print en feature */  
    stijldeclaraties  
}
```

Voor het combineren is er behalve and (beide voorwaarden moeten waar zijn) en de komma die of betekent (het is genoeg als een van de voorwaarden waar is), ook het sleutelwoord not. Dit betekent dat de media query alleen wordt uitgevoerd als wat volgt niet waar is.



Afbeelding 10.2 Op mediaqueri.es zijn talloze voorbeelden van ontwerpen te bekijken.

query heeft de volgende algemene vorm (let op het sleutelwoord `and` en de haakjes om de expressie):

```
@media apparaat and (media feature) {  
    selector {  
        declaraties;  
    }  
}
```

- Elke media query begint met `@media`.
- Waarden voor `apparaat` zijn `screen` (beeldscherm), `print` (printer) en `speech` (spraak). Als u niets instelt is de waarde `all`.
- Waarden voor `media feature` zijn bijvoorbeeld `min-width` of `orientation` (zie hierna).
- Het sleutelwoord `and` wordt gebruikt om expressies te combineren.
- Alle inhoud van de media query staat tussen accolades.
- Elke stijlregel heeft de gebruikelijke accolades.
- Media queries voor verschillende apparaten worden gescheiden door een komma. In feite is de komma een `or`-operator:

```
@media screen and (feature), print and (feature) /* of screen en feature, of print en feature */  
{  
    stijldeclaraties  
}
```

Voor het combineren is er behalve `and` (beide voorwaarden moeten waar zijn) en de komma die `or` betekent (het is genoeg als een van de voorwaarden waar is), ook het sleutelwoord `not`. Dit betekent dat de media query alleen wordt uitgevoerd als wat volgt niet waar is.

MIT



Afbeelding 10.2 Op mediaqueri.es zijn talloze voorbeelden van ontwerpen te bekijken.

De aanbevolen methode is om de media queries op te nemen in dezelfde style-sheet als de andere opmaak. In die stylesheet zouden alle media queries in een blok bij elkaar kunnen staan, maar dat maakt het werk niet overzichtelijker. Een aanbevolen methode, in combinatie met mobile first, is om de media query op te nemen direct na het element dat wordt aangepast, dus na de basisstijlen van dat element. Bij mobile first betekent deze werkwijze dat in de weergave voor kleine schermen alle basisopmaak van het element beschikbaar is. In de media query hoeven alleen de aanpassingen voor een groter scherm te worden opgenomen.

Een pagina heeft bijvoorbeeld een header, hoofdinhoud met een artikel en een zijkalk en een footer. Op een smartphone staan die onderdelen vanzelf onder elkaar en dat is goed. Maar op een tablet die in landschapstand 1024px breed is, moeten de hoofdinhoud en zijkalk naast elkaar staan en dat gaat niet vanzelf, want het zijn allemaal blokboxen.

```
<header> </header>
<main>
  <article> </article>
  <aside> </aside>
</main>
<footer> </footer>
```

De eenvoudigste oplossing is van `<main>` een gridlay-out te maken op schermen van 1024px en groter:

```
@media screen and (min-width: 1024px) {
  main {
    display: grid;
    grid-template-columns: 66% 34%;
  }
}
```

Gridlay-out worden verderop uitgebreid besproken, maar hier staat:

- Is het medium een scherm en is dat scherm minimaal 1024px breed?
- Zo ja, selecteer het element `<main>`:
 - stel de eigenschap `display` in op `grid`;
 - maak twee kolommen waarvan de eerste 66% en de tweede 34% van de breedte krijgt.
- Zo nee, negeer dan alle declaraties in dit `@media`-blok.



Relatieve breekpunten

In het voorbeeld is het breekpunt gegeven in pixels, maar het kan net zo goed (of beter) met relatieve lengten in em of rem. Daarmee wordt het breekpunt gekoppeld aan de grootte van tekst, waardoor er balans blijft tussen de lay-out en de tekstgrootte.

Schermresolutie

Op schermen met een hoge resolutie (een hoge device pixel ratio, DPR) kunnen afbeeldingen met meer detail worden bekijken. Een media query voor de resolutie zorgt ervoor dat zo'n gedetailleerde afbeelding alleen wordt geladen op een apparaat met een scherm waarop dat zin heeft. Het criterium heet **resolution** (min- of max-) en waarden kunnen in diverse eenheden worden gegeven:

- dpi dots per inch
- dpcm dots per centimeter
- dppx dots per pixel

De eenheid dppx is te vinden in de CSS-module *Values and units level 3*. Nu komt 1dppx overeen met 96dpi, de standaardresolutie van een beeldscherm. Op zo'n scherm is het verspilling van bandbreedte om een foto van 192dpi te laden, maar er zijn zeker apparaten die wel die resolutie kunnen weergeven. Met een media query kan worden bepaald welke afbeelding wordt geladen:

```
header {  
    background-image: url('headerafb-laag.jpg');  
}  
@media screen and (min-resolution: 2dppx) {  
    header {  
        background-image: url('headerafb-hoog.jpg')  
    }  
}
```

Portret of landschap

Met orientation wordt gedetecteerd wat de stand van het venster is: staand of liggend. De mogelijke waarden zijn **portrait** en **landscape**. Het biedt een wat ruimere marge dan selecteren op een bepaalde breedte, maar dat kan voor een ontwerp voldoende zijn. Een voor de hand liggende toepassing is dat de navigatie wordt omgezet als wordt gewisseld van een horizontale weergave (landschap) naar een verticale (portret).

```

@media screen and (orientation: landscape) {
    nav ul li {
        float: left;
    }
}

@media screen and (orientation: portrait) {
    nav ul li {
        float: none;
    }
}

```

Beeldverhouding

Wanneer alleen de beeldverhouding belangrijk is, kan aspect-ratio (browservenster) worden gebruikt. Deze kan de voorvoegsels min- en max- krijgen. De beeldverhouding is bijvoorbeeld 4:3 (1280 x 960) of breedbeeld 16:9, 16:10 enzovoort. Bedenk dat de verhouding exact moet kloppen, want anders wordt de bijbehorende stijlregel niet toegepast. Een media query voor breedbeeld is bijvoorbeeld:

```

@media screen and (aspect-ratio: 16/9) {
    ...
}

```



Nog meer mogelijkheden

Op www.w3.org/TR/css3-mediaqueries/ zijn in de specificatie alle mogelijkheden van media queries vinden.

Weergave op mobiele schermen: de metatag viewport

Een webpagina is niet zomaar goed te bekijken op een mobiel apparaat. Dat komt doordat de breedte waarin de pagina wordt opgebouwd een andere breedte is dan die van het scherm. Met inzoomen en zijwaarts scrollen is de inhoud leesbaar te maken, maar een prettige gebruikerservaring is het zeker niet.

De oorzaak is een combinatie van de CSS-eigenschap width en de viewport van het mobiele apparaat. De viewport is op een desktopbrowser gelijk aan het browservenster. Wordt het browservenster groter of kleiner gemaakt, dan wordt ook de viewport groter of kleiner. Op een mobiel kan dat niet. De viewport van een browser op een telefoon heeft vaste afmetingen. Alleen zijn die niet in elke browser hetzelfde en om het nog lastiger te maken heeft een mobiele browser twee viewports: een zichtbare viewport en een lay-outviewport.

De zichtbare viewport is het scherm van het mobiele apparaat. De lay-outviewport is bij de meeste apparaten veel groter. Hoe groot precies, dat verschilt per browser. De breedte varieert van zo'n 800 tot 980 pixels. Stelt u zich de lay-outviewport voor als een grote afbeelding die wordt afgedekt door een ondoorzichtige laag met een venster. De zichtbare viewport is het venster waardoorheen een deel van die onderliggende afbeelding zichtbaar is.

De CSS-eigenschap `width` is gerelateerd aan de lay-outviewport. Een pagina met `width: 100%` is dan 980 pixels breed, terwijl het scherm (de zichtbare viewport) maar 375 pixels breed is (portretstand iPhone X). Die meldt overigens 1125 pixels breed te zijn (3×375), maar dat komt doordat op een iPhone X-scherm horizontaal drie hardwarepixels in een CSS-pixel gaan (zie hierna over CSS-pixels).

Naast de CSS-eigenschap `width` bestaat de eigenschap `device-width`. Die komt overeen met de breedte van het telefoonscherm en dat is dan ook de afmeting die de webpagina zou moeten hebben.

Zo komen we bij de metatag `viewport`. Door de `width` van `viewport` uitdrukkelijk in te stellen op de `device-width`, wordt een `width` van 100% hetzelfde als de breedte van het venster, bij de iPhone X is dat 375 pixels.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

De toevoeging `initial-scale=1` zorgt ervoor dat de webpagina niet wordt geschaald om hem in het venster te laten passen, dus 1 CSS-pixel is gelijk aan 1 pixel in de viewport. Als de pagina niet flexibel is (een vaste breedte heeft), is het gevolg van de metatag maar half positief: de tekst is weliswaar leesbaar, want er wordt niet ingezoomd. Er moet daardoor echter ook horizontaal worden gescrold; de pagina is op de achtergrond immers veel breder dan het scherm (die 800 tot 980 pixels van de lay-outviewport).



Device-features afgekeurd

In de huidige versie van de module voor media queries (level 4) worden alle device-features afgekeurd: `device-width`, `device-height` en `device-aspect-ratio`. Of dat ook echt gebeurt is nog niet te zeggen. We zouden ook niet weten wat dat voor de `viewport`-declaratie betekent. Houd het daarom voorlopig op wat hier is beschreven.

CSS-pixels en hardwarepixels (device pixels)

Behalve twee viewports zijn er ook twee soorten pixels. De CSS-pixel is een abstracte eenheid waarvan de grootte (mede) wordt bepaald door de kijkafstand tot het scherm. De hardwarepixel is de echte beeldpunt op het scherm. De CSS-pixel is de basis voor de weergave op het scherm en dus voor uw CSS-declaraties. (Mobiele) schermen krijgen echter steeds hogere resoluties. Om meer punten op dezelfde oppervlakte te kunnen plaatsen, moeten de punten kleiner worden. Bij een verhouding tussen de CSS-pixel en de hardwarepixel van 1 : 1 zouden elementen op de webpagina miniaturen worden, en onleesbaar. Een voorbeeld: de iPhone X met een schermbreedte van 375 pixels heeft in werkelijkheid 1125 pixels beschikbaar. De *device pixel ratio* (DPR) van dit apparaat is 3. Dat betekent dat er drie hardwarepixels in een CSS-pixel gaan (of eigenlijk negen: drie breed en drie hoog).

Nu maakt het voor de CSS-opmaak eigenlijk niet uit hoe het met die pixels zit. Een width van 375px in de stylesheet is ook 375px in de weergave. Voor afbeeldingen doet het er echter des te meer toe, want op diezelfde iPhone kan nu een afbeelding van 1125 pixels worden gebruikt. Deze worden getoond binnen de 375 CSS-pixels en dat levert een veel scherper plaatje op. Googel voor meer informatie de termen `high resolution images`, `viewport device-width` en `device pixel ratio`. Een aanbevolen artikel is medium.com/@pnowelldesign/pixel-density-demystified-a4db63ba2922.

Lay-out: positionering

De manier waarop de boxen van het boxmodel en het weergavemodel worden geordend in het browservenster wordt bepaald door het positionerings-schema, relaties tussen de elementen in de HTML-boomstructuur en externe informatie zoals schermgrootte/venstergrootte, eigen afmetingen van afbeeldingen enzovoort. In deze paragraaf kijken we naar het positioneringsschema, te vinden in *CSS Positioned Layout level 3* (www.w3.org/TR/css-position-3/).

Het omvattende blok

Een belangrijk begrip bij positionering is het omvattende blok (*containing block*). Dit is het blok waarin andere boxen zijn geplaatst en wordt vaak *container* genoemd. Het is dus niet de box die een element zelf genereert. Elke box krijgt een positie in een omvattend blok, maar is daarin niet opgesloten; een box kan overlappen en of dat gebeurt is afhankelijk van de eigenschap `overflow` (zie hoofdstuk 9, de paragraaf *Overlopende inhoud*).

Met de volgende richtlijnen kunt u in de meeste gevallen uit de voeten:

- Het omvattende blok van het rootelement <html> is het browservenster (beter: de *viewport*).
- Het omvattende blok van andere elementen is de eerste voorouder op blokniveau.
- Het omvattende blok van een absoluut geïnlineerd element is de meest nabije voorouder met een andere waarde voor position dan static (meestal wordt relative gebruikt) en als die er niet is wordt het browservenster het containing blok. Om een box absoluut te positioneren in een zelfgekozen blok, geeft u dat blok dus position: relative.

Positioneringsschema's

Er zijn er drie positioneringsschema's:

- Normale flow De volgorde waarin de elementen in de HTML staan, waarbij elementen worden geplaatst afhankelijk van hun display-waarde (block of inline) en daarbij kunnen worden verschoven (relatieve positionering).
- Floats Een float is een 'zwevende' box die eerst volgens de normale flow wordt geplaatst en daar vervolgens wordt uitgehaald en links of rechts wordt geplaatst. Andere inhoud kan eromheen lopen (of juist niet; dat is een instelbare mogelijkheid).
- Absolute positionering Een box wordt volledig uit de normale flow gehaald en krijgt een positie toeewezen in zijn omvattende blok.

De eigenschap position

De plaatsing van een box wordt bepaald door de eigenschap position. De mogelijke waarden zijn:

- static (de standaardwaarde), de box volgt de normale flow
- relative
- absolute
- fixed
- sticky

```
element {  
    position: relative; /* absolute | fixed | sticky */  
}
```

Verschuiven met top, right, bottom en left

Wanneer een element wordt gepositioneerd (`position` heeft niet de waarde `static`), wordt de plaats of de verschuiving aangegeven met de volgende vier eigenschappen:

- `top` bovenrand
- `bottom` onderrand
- `right` rechterrand
- `left` linkerrand

Mogelijke waarden zijn `auto`, een lengte of een percentage.

Verschuiven met logische waarden

In plaats van `top`, `bottom`, `left` en `right` kunnen de nieuwe logische waarden voor verschuiven worden gebruikt. Deze zijn gekoppeld aan de schrijfrichting. (Zie ook hoofdstuk 9, de paragraaf *De toekomst: logische waarden*).

Logische waarde	Fysieke waarde
<code>offset-before</code>	<code>top</code>
<code>offset-after</code>	<code>bottom</code>
<code>offset-start</code>	<code>left</code>
<code>offset-end</code>	<code>right</code>

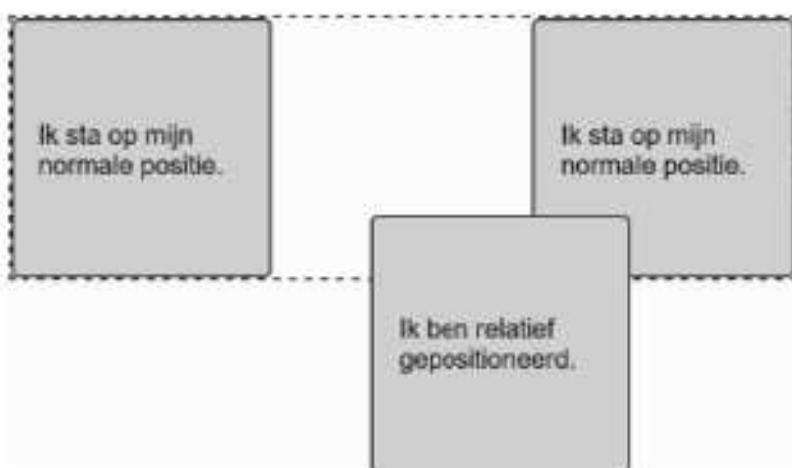
Position: static

Bij `position: static` zijn alle verschuivingswaarden `auto`, de box volgt de normale flow. Dit is de standaardwaarde.

Position: relative

Bij relatieve positionering wordt een box verplaatst ten opzichte van zijn positie in de normale flow. De box wordt eerst op zijn normale plek gezet en daarna verschoven. Een relatief geplaatste box heeft geen invloed op omringende boxen. Daardoor kunnen ze elkaar overlappen of afdekken. Relatief positioneren heeft geen invloed op de grootte van de box en de ruimte die deze inneemt. Een element wordt relatief geplaatst met `position: relative`.

Met de eigenschap `left` en een positieve waarde wordt een box naar rechts verplaatst en met `right` naar links. Negatieve waarden zijn ook mogelijk; dan gebeurt het omgekeerde. De verplaatsing naar de ene kant is altijd het omgekeerde van die naar de andere kant. Bij `left: 50px` geldt automatisch



Afbeelding 10.3 De tweede box twee is relatief gepositioneerd. De hoek linksboven is 150px omlaag en 75px naar rechts verschoven. De ruimte die de box oorspronkelijk innam, is nog steeds bezet: de derde box schuift niet omhoog (10_03.html).

right: -50px. Bij verplaatsing omlaag en omhoog, met top en bottom, geldt eveneens dat de ene waarde altijd het omgekeerde is van de andere.

Twee voorbeelden:

- left: 50px verplaats 50 pixels naar rechts.
- right: -50px verplaats -50 pixels naar links, wat hetzelfde is als 50 pixels naar rechts.

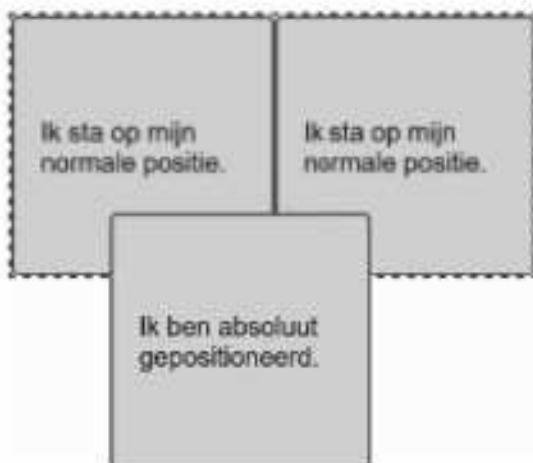
Een relatief gepositioneerde box is automatisch een nieuw omvattend blok voor absoluut gepositioneerde afstammelingen. Dat is een heel handige eigenschap om binnen boxen op een eenvoudige manier onderdelen een vaste plek te geven. U hoeft namelijk alleen rekening te houden met de relatieve box. De relatief gepositioneerde box hoeft niet te worden verplaatst; het werkt ook zonder verschuivingswaarden.

Voorbeeld: als u de box met de hoofdinhoudblock van de pagina relatief positioneert (zonder verschuiving) en daarin een andere box absoluut positioneert, volgt de hoofdinhoudblock de normale loop van de pagina en blijft de daarbinnen absoluut gepositioneerde box altijd op zijn plaats.

Position: absolute

In de normale flow, bij relatieve positionering en bij floats is de plaats die het element in de HTML-code inneemt het uitgangspunt voor de plaats van de elementen in de weergave. Bij de normale flow spreekt dat vanzelf, maar ook relatief gepositioneerde elementen en floats worden geplaatst ten opzichte van hun normale positie.

Bij absolute positionering is er geen normale positie. De plaats van de box (het element) wordt volledig bepaald door het omvattende blok en de coördinaten in de stijlregel (met top, right, bottom en left). Een absoluut geplaatste box wordt uit de normale flow gehaald en heeft geen invloed op wat erna komt. De box trekt zich ook niets aan van wat er op de aangegeven positie staat; als daar een andere box staat, wordt die overlaat, net als bij relatieve positionering. Bij geplaatste elementen onderling hangt overlapping af van de volgorde van de elementen in het bronbestand. Lager in de code is hoger in de stapel.



Afbeelding 10.4 De tweede box is absoluut geplaatst. De hoek linksboven staat op 75px van links en 150px van boven. Het omvattende blok is een container om de drie boxen. De vrijgekomen ruimte is ingenomen door de derde box (10_04.html).

Padding op het containing blok heeft geen invloed op de absoluut geplaatste box. Marge op die box heeft wel invloed op de plaatsing ten opzichte van het containing blok.



Afbeelding 10.5 Het omvattende blok heeft padding en is daardoor groter, maar de absoluut geplaatste box blijft op dezelfde plek staan (10_04.html).

Bij absolute positionering is er geen normale positie. De plaats van de box (het element) wordt volledig bepaald door het omvattende blok en de coördinaten in de stijlregel (met top, right, bottom en left). Een absoluut geplaatste box wordt uit de normale flow gehaald en heeft geen invloed op wat erna komt. De box trekt zich ook niets aan van wat er op de aangegeven positie staat; als daar een andere box staat, wordt die overlaat, net als bij relatieve positionering. Bij geplaatste elementen onderling hangt overlapping af van de volgorde van de elementen in het bronbestand. Lager in de code is hoger in de stapel.



Afbeelding 10.4 De tweede box is absoluut geplaatst. De hoek linksboven staat op 75px van links en 150px van boven. Het omvattende blok is een container om de drie boxen. De vrijgekomen ruimte is ingenomen door de derde box (10_04.html).

Padding op het containing blok heeft geen invloed op de absoluut geplaatste box. Marge op die box heeft wel invloed op de plaatsing ten opzichte van het containing blok.



Afbeelding 10.5 Het omvattende blok heeft padding en is daardoor groter, maar de absoluut geplaatste box blijft op dezelfde plek staan (10_04.html).

```
right: 0;
```

Ook een header kan boven aan de pagina worden vastgezet, waarbij de andere inhoud eronderdoor schuift. Dan moet de andere inhoud minimaal de hoogte van de header als margin-top hebben, anders wordt bij het laden van de pagina de bovenste inhoud afgedekt door de header.

Position: sticky

Een betrekkelijk nieuwe positiewaarde is sticky. De browserondersteuning is nog onvolledig en dat komt vooral doordat sticky nog niet volledig is uitgewerkt in de specificatie; dat maakt het lastig om het in browsers te programmeren.

Een box met position: sticky wordt geplaatst zoals een relatief geplaatste box, maar het vastzetpunt (de offset) wordt gekoppeld aan de eerste voorouder die een schuifbalk mag maken. Als die voorouder er niet is, is het browservenster (de viewport) het ankerpunt. Als door scrollen de ingestelde positie wordt bereikt, blijft de box op die positie ‘plakken’ totdat zijn omvattende blok uit beeld scrollt. (Een box mag een schuifbalk maken als de eigenschap overflow de waarde auto of scroll heeft, zie ook hoofdstuk 9, de paragraaf *Overlopende inhoud*.)

In het voorbeeld zijn de koppen van artikelen sticky, waardoor elke kop zichtbaar blijft terwijl de bijbehorende tekst wordt gescrold. De samenvatting van de code is:

```
<aside>
  <article>
    <h2>Prikkende kop</h2>
    <p>Lorem ipsum ...</p>
  </article>
  <article>
    <h2>Kog een kop</h2>
    <p>Lorem ipsum ...</p>
  </article>
</aside>
/* CSS */
aside {
  background-color: rgba(200, 100, 100, .5);
  height: 400px;
  overflow: auto;
  padding: 0 1em;
```

```

width: 300px;
}

h2 {
background-color: #f0f0f0;
margin: 0;
position: sticky;
top: 0;
}

article {
margin-top: 2em;
}

article:last-of-type {
margin-bottom: 100vh;
}

```

Prikkelende kop

sapiente similiqe
temporibus vero!

Nog een kop

Lorem ipsum dolor sit amet, consectetur adipisciing elit. Accusamus dicta ipsam itaque optio quas sit voluptatem! Asperiores cumque dolorum itaque iusto optio possimus quam reprehenderit rerum, sapiente similiqe temporibus vero!

Prikkelende kop

Lorem ipsum dolor sit amet, consectetur adipisciing elit. Accusamus dicta ipsam itaque optio quas sit voluptatem! Asperiores cumque dolorum itaque iusto optio possimus quam reprehenderit rerum, sapiente similiqe temporibus vero!

Nog een kop

Afbeelding 10.7 Rechts de beginsituatie, links is gescrold. De tekst schuift onder de kop door. De kop schuift mee uit beeld met het einde van het artikel (10_07.html).

Analyse van de code

Hier is de `<aside>` de container voor de artikelen. Om te kunnen scrollen is er meer inhoud geplaatst dan er in de container past en de schuifbalk wordt automatisch toegevoegd dankzij de eigenschap `overflow: auto`. De artikelen hebben een bovenmarge van 2em, zodat goed is te zien dat de kop eerst volgens de normale flow boven in de box van het `<article>` wordt geplaatst.

Als het scrollen begint, schuift het `<article>` inclusief de kop omhoog in de `<aside>`. Als de sticky kop de bovenkant bereikt van de `<aside>`, blijft die daar plakken op de ingestelde positie `top: 0`. De `<aside>` is namelijk de eerste

voorouder die een schuifbalk mag maken. Experimenteer zelf met verschillende offsetwaarden van de kop.

Het laatste artikel heeft een grote ondermarge, zodat goed is te zien dat de kop uit beeld verdwijnt met het artikel; de kop wordt dus niet weggeduwd door de volgende kop.

Float en clear

Een float of zwevende box is een box die links of rechts op de huidige regel wordt geplaatst. Het bijzondere van een float is dat inhoud er omheen kan lopen. Dat kan overigens ook worden voorkomen met de eigenschap `clear`. Die komt niet op de float zelf, maar op het blok erna.

Met de stijlregel `float: left` of `float: right` wordt een box naar links of rechts verschoven tot aan de rand van het omvattende blok of tot de rand van een andere float. Bij een float naar links loopt de inhoud langs de rechterkant en bij een float naar rechts loopt de inhoud langs de linkerkant. De inhoud begint op dezelfde regel als de float.

Floats zijn bedacht om tekstromloop mogelijk te maken, zoals in gedrukte media. Een veelgebruikte toepassing is daarom een afbeelding floaten en tekst eromheen laten lopen. Daarvoor wordt in de HTML een afbeelding in een `alinea` gezet. In de CSS krijgt `img` de eigenschap `float: left`.

```
<article>
  <h2>Float</h2>
  <p> ... tekst ... </p>
</article>
/* CSS */
img {
  float: left;
}
```

Het element `` hoeft niet per se aan het begin of eind van de `alinea` te staan; tussen twee zinnen in of elke op andere plaats kan ook. De tekstromloop begint altijd op de regel waar de afbeelding terechtkomt.

Een gefloateerde afbeelding (maar dit geldt eveneens voor ook elke andere gefloateerde box) wordt uit de elementflow gehaald. Het gevolg daarvan is dat andere blokboxen geen rekening houden met de float. Daardoor zal een volgende blokbox omhoog schuiven tot die een niet-gefloateerde blokbox tegenkomt, zoals in de volgende afbeelding is te zien.

Float: afbeelding naar links	Float: afbeelding naar rechts
 <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ad alias animi aperiam debitis earum, eos et, laborum magni maiores molestiae nemo quia quibusdam reiciendis ut voluptate. Consequatur esque et sint. Dolorem, nostrum, vero!</p>	 <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ad alias animi aperiam debitis earum, eos et, laborum magni maiores molestiae nemo quia quibusdam reiciendis ut voluptate. Consequatur esque et sint. Dolorem, nostrum, vero!</p>

Afbeelding 10.8 Tekst rondom een afbeelding laten lopen met float (10_08.html).

Float: afbeelding naar links	Float: afbeelding naar rechts
 <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ad alias animi aperiam debitis earum, eos et, laborum magni maiores molestiae nemo quia quibusdam reiciendis ut voluptate. Consequatur esque et sint. Dolorem, nostrum, vero!</p>	 <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ad alias animi aperiam debitis earum, eos et, laborum magni maiores molestiae nemo quia quibusdam reiciendis ut voluptate. Consequatur esque et sint. Dolorem, nostrum, vero!</p>

Afbeelding 10.9 Het probleem met floats. De eerste afbeelding loopt zijn artikelcontainer uit (de kaderlijn). De blokbox van het tweede artikel negeert de float en sluit aan op de eerste blokbox. Ook de tweede afbeelding loopt zijn artikelcontainer uit (10_09.html).

Om te voorkomen dat een volgende blokbox door de float heen loopt, krijgt die box de eigenschap clear. Met clear: left of clear: right wordt aangegeven dat aan de genoemde zijde geen float mag staan. Met clear: both mag naast de box nooit een float staan. De standaardwaarde is clear: none, wat betekent dat floats aan beide zijden zijn toegestaan.

Float: afbeelding naar links

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ad alias animis speriam debitis earum, eos et, laborum magni malores molestiae nemo quia quibusdam reiciende ut voluptate. Consequatur esque et sint. Dolorem, nostrum, vero!

Float: afbeelding naar rechts

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ad alias animis speriam debitis earum, eos et, laborum magni malores molestiae nemo quia quibusdam reiciende ut voluptate. Consequatur esque et sint. Dolorem, nostrum, vero!



Afbeelding 10.10 Met `clear: left` op het tweede artikel komt het onder de gefloate afbeelding uit het eerste artikel te staan (10_10.html).

Float als hulpmiddel voor lay-outs

Floats zijn jarenlang gebruikt voor het maken van paginalay-outs met meerdere flexibele kolommen. Intussen zijn daar andere en betere hulpmiddelen voor, zoals gridlay-out en flexbox. In deze vijfde editie van het handboek wordt daarom voor het eerst niet meer diep ingegaan op het maken van complete lay-outs met float. Basiskennis is wel handig als floats voor een (onderdeel van een) project toch een handige oplossing zijn.

In de paragraaf hiervoor is een afbeelding gefloat die in een alinea staat, maar (meerdere) blokboxen kunnen ook worden gefloat. Bekijk de volgende code.

```
<div class="container">
<h2>Floats naast elkaar</h2>
<div>Float</div>
<div>Float</div>
<div>Float</div>
<div>Float</div>
</div>
/* CSS */
.container {
  background-color: #ccc;
  padding: 1em 0 1em 1em;
}
h2 {
  margin-top: 0;
}
```

```
div > div {
    float: left;
    background-color: #ddd;
    height: 300px;
    margin: 0 1em 0 0;
    width: 150px;
}
```

FLOATS NAAST ELKAAR



Afbeelding 10.11 Het resultaat van de code, de achtergrondkleur van de container is donkerder dan die van de floats (10_11.html).

De code maakt een container en daarin staan een kop en vier blokboxen (`<div>`). De container heeft een achtergrondkleur en padding. De div's hebben vaste afmetingen en de eigenschap `float: left`. In de afbeelding is te zien dat er zoveel gefloate boxen naast elkaar komen te staan als er in de container passen. Is er te weinig ruimte is, dan gaat de laatste float verder op de volgende regel. Als het browservenster smaller wordt gemaakt, passen er minder floats naast elkaar en uiteindelijk staan alle floats onder elkaar. Flexibele lay-out!

Maar er is wel een probleem en eigenlijk zijn het twee. Het eerste is nu al te zien: de container omvat de floats wel horizontaal, maar niet verticaal, zie de achtergrondkleur die niet doorloopt onder de floats. Het tweede probleem is daar een gevolg van en dat wordt zichtbaar als na de floats een blokbox wordt toegevoegd en het scherm smaller of breder wordt gemaakt.

```
<div class="container">
    <h2>Floats naast elkaar</h2>
    <div class="float">Float</div>
    <div class="float">Float</div>
    <div class="float">Float</div>
    <div class="float">Float</div>
```

```
</div>
<div>
  <h2>Floats naast elkaar</h2>
  <p>Deze tekst loopt dwars door de voorgaande float heen.</p>
</div>
```

Floats naast elkaar

Float Float Float

Float

Floats in de kluts

Deze tekst loopt dwars door de voorgaande float heen.

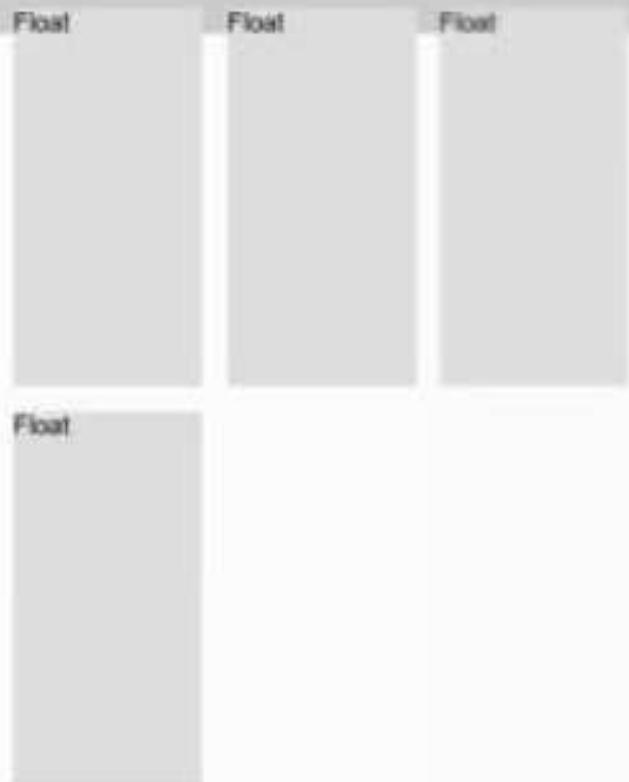
Afbeelding 10.12 De container omvat de floats niet en daardoor kan de volgende blokbox erdoorheen lopen (10_12.html).

Met `clear: left` kan worden voorkomen dat er naast de toegevoegde div een float komt te staan.

```
.clear {
  clear: left;
}
```

Nu komt de toegevoegde box komen wel onder de laatste gefloate box te staan, maar de container van de floats wordt er niet hoger door. Het probleem met de achtergrondkleur is dus niet opgelost. De echte oplossing zit in het maken van een nieuwe block formatting context.

FLOATS NAAST ELKAAR



FLOATS IN DE KLUTS

Deze tekst loopt niet meer door de voorgaande float heen.

Afbeelding 10.13 Het probleem is half opgelost (10_13.html).

Block formatting context

Het probleem van de uitstekende floats is op te lossen door van de box met de floats een nieuwe *block formatting context* te maken. Een block formatting context (BFC) is een soort privéterrein binnen de normale flow, met een hek om de inhoud. Een element dat een BFC maakt, omvat alle elementen die erin zitten; het voorkomt dat floats buiten de container kunnen staan. Het voor komt overigens ook het samenvoegen van marges.

Er zijn verschillende manieren om van een box een BFC te maken. Een lang bestaande methode is door de eigenschap `overflow` van de box een andere waarde te geven dan `visible`; meestal is `auto` de veiligste keus (zie ook hoofdstuk 9, de paragraaf *Overlopende inhoud*). Toch kunnen er met deze methode nog ongewenste schuifbalken verschijnen. Dat is overigens geen fout; dat is precies wat `overflow` moet doen.

De moderne methode om een BFC te maken is met `display: flow-root`; deze waarde is eerder besproken bij De eigenschap `display`. Het resultaat is dat de container de floats omsluit. Zie caniuse.com voor browserondersteuning.

De clear op de laatste div wordt verwijderd. In plaats daarvan komt op de eerste <div> de eigenschap display: flow-root. De eerste is immers de container van de gefloateerde div's.

```
.container {
    display: flow-root;
}
```

Een BFC wordt ook gemaakt door position: absolute of position: fixed in te stellen. Ook de waarden display: inline-block, display: table-cell en display: table-caption maken een BFC. Omdat bij het maken van

Floated next to each other

Float Float Float Float



Floated in the mess

This text does not run through the previous float.

Image 10.14 *Today the container of the floats creates a new BFC and surrounds the floats now really (10_14.html).*

een HTML-tablet de boxen die van de elementen worden gemaakt automatisch deze display-waarden hebben, zal elke cel van een tabel een BFC zijn. Tot slot maakt ook column-span: all een BFC. Deze eigenschap hoort bij meerkolomslayout en dat onderwerp komt hierna aan bod.

As flow-root does not work: clearfix

Als een voor uw publiek belangrijke browser de waarde flow-root niet ondersteunt, kunt u terugvallen op de oude methode: declearfix. Daarvoor wordt de volgende CSS-code ingesteld op de container van de floats.

```
.container:after {
    content: "";
```

De clear op de laatste div wordt verwijderd. In plaats daarvan komt op de eerste <div> de eigenschap display: flow-root. De eerste is immers de container van de gefloate div's.

```
.container {
    display: flow-root;
}
```

Een BFC wordt ook gemaakt door position: absolute of position: fixed in te stellen. Ook de waarden display: inline-block, display: table-cell en display: table-caption maken een BFC. Omdat bij het maken van

Floats naast elkaar

Float Float Float Float



Floats in de kluts

Deze tekst loopt niet meer door de voorgaande float heen.

Afbeelding 10.14 Tad! De container van de floats maakt een nieuwe BFC en omsluit de floats nu echt (10_14.html).

een HTML-tabel de boxen die van de elementen worden gemaakt automatisch deze display-waarden hebben, zal elke cel van een tabel een BFC zijn. Tot slot maakt ook column-span: all een BFC. Deze eigenschap hoort bij meerkolomslayout en dat onderwerp komt hierna aan bod.

Als flow-root niet werkt: clearfix

Als een voor uw publiek belangrijke browser de waarde flow-root niet ondersteunt, kunt u terugvallen op de oude methode: de clearfix. Daarvoor wordt de volgende CSS-code ingesteld op de container van de floats.

```
.container:after {
    content: "";
```

De clear op de laatste div wordt verwijderd. In plaats daarvan komt op de eerste <div> de eigenschap display: flow-root. De eerste is immers de container van de gefloate div's.

```
.container {
    display: flow-root;
}
```

Een BFC wordt ook gemaakt door position: absolute of position: fixed in te stellen. Ook de waarden display: inline-block, display: table-cell en display: table-caption maken een BFC. Omdat bij het maken van

Floats naast elkaar

Float Float Float Float



Floats in de kluts

Deze tekst loopt niet meer door de voorgaande float heen.

Afbeelding 10.14 Tad! De container van de floats maakt een nieuwe BFC en omsluit de floats nu echt (10_14.html).

een HTML-tabel de boxen die van de elementen worden gemaakt automatisch deze display-waarden hebben, zal elke cel van een tabel een BFC zijn. Tot slot maakt ook column-span: all een BFC. Deze eigenschap hoort bij meerkolomslayout en dat onderwerp komt hierna aan bod.

Als flow-root niet werkt: clearfix

Als een voor uw publiek belangrijke browser de waarde flow-root niet ondersteunt, kunt u terugvallen op de oude methode: de clearfix. Daarvoor wordt de volgende CSS-code ingesteld op de container van de floats.

```
.container:after {
    content: "";
```

Er is een CSS-techniek waarmee inhoud over meerdere kolommen kan lopen: [CSS Multi-column layout](#). Daarbij kunnen een vast aantal kolommen en de gewenste breedte van de kolommen worden ingesteld. Er worden zo veel kolommen gemaakt als er in de container passen.

De inhoud van de kolommen kan van alles zijn: tekst, afbeeldingen, lijsten, containers met daarin weer tekst en afbeeldingen enzovoort.

Binnen de kolommen wordt de normale flow van elementen gevolgd. Multi-column is niet geschikt om complete pagina's op te maken. Het is wel geschikt voor het in kolommen weergeven van onderdelen van een pagina, bijvoorbeeld artikelen (met afbeeldingen), een paginafooter met veel informatie of productkaarten. Ook een gerubriceerde lijst van producten of iets dat daarop lijkt kan een goede toepassing zijn.

Afbeelding 10.15 Met een waarde voor column-count wordt inhoud in een vast aantal kolommen geplaatst. Met padding op article krijgt de tekst wat lucht (10_15.html).

Deze kolommen worden zo breed als de ruimte in de container of het browservenster toestaat. Ook als het venster smaller wordt, blijven het twee kolommen. (Met een media query kan het aantal kolommen bij een bepaalde schermafmeting worden aangepast, zie de paragraaf *Responsive design*.)

Voor meer flexibiliteit kan de gewenste kolombreedte worden ingesteld.

```
article {
    column-width: 175px;
}
```

Het aantal en de breedte van kolommen hangt nu af van de ruimte in de container. Als de container breder wordt, zullen de kolommen ook breder worden, totdat er weer een kolom van 200px naast past. De breedte van column-width is een minimale breedte.

Er is een CSS-techniek waarmee inhoud over meerdere kolommen kan lopen: [CSS Multi-column layout](#). Daarbij kunnen een vast aantal kolommen en de gewenste breedte van de kolommen worden ingesteld. Er worden zo veel kolommen gemaakt als er in de container passen.

De inhoud van de kolommen kan van alles zijn: tekst, afbeeldingen, lijsten, containers met daarin weer tekst en afbeeldingen enzovoort. Binnen de kolommen wordt de normale flow van elementen gevolgd. Multi-column is niet geschikt om complete pagina's op te maken. Het is wel geschikt

voor het in kolommen weergeven van onderdelen van een pagina, bijvoorbeeld artikelen (met afbeeldingen), een paginafooter met veel informatie of productkaarten. Ook een gerubriceerde lijst van producten of iets dat daarop lijkt kan een goede toepassing zijn.

Afbeelding 10.16 Hier zijn de kolommen minimaal 200px breed (10_16.html).

Met de eigenschap `columns` werkt het twee kanten op. De volgende code maakt kolommen nooit smaller dan 200px en het aantal wordt nooit groter dan twee. Test dit zelf in de browser.

```
p {  
    columns: 175px 2;  
}
```

De volgorde van de waarden maakt niet uit. Het aantal is altijd een integer en de breedte is altijd een lengte met een eenheid, dus dat kan niet fout gaan. Als maar één waarde wordt gegeven, wordt de andere auto.

Marges worden niet samengevoegd

In de twee voorgaande afbeeldingen ziet de tekstloop er niet fraai uit. De eerste kolom begint lager dan de tweede en aan de onderkant lijnt de tekst ook niet. De oorzaak hiervan is een paar pagina's eerder uitgelegd: de meerkolomslay-out maakt een nieuwe block formatting context (BFC) en een van kenmerken daarvan is dat marges niet worden samengevoegd.

De oplossing is het aanpassen van de boven- en ondermarge van de inhoud, in dit geval de alinea's. Als er maar één alinea is kunnen ze beide op nul worden ingesteld, maar in de praktijk komt het erop neer dat de instelling van de marges wat preciezer moet gebeuren. In de volgende afbeelding is een mogelijk oplossing toegepast. Van elke alinea is de marge boven en onder nul, maar elke alinea die wordt voorafgegaan door een andere alinea heeft wel een bovenmarge. Met padding op de container krijgt de tekst wat ruimte.

```
<article>  
  <p> ... </p>  
  <p> ... </p>  
</article>  
/* CSS */  
article {  
    padding: 1em;  
}  
p {  
    margin: 0;  
}  
p + p {  
    margin-top: 1em;  
}
```

Er is een CSS-techniek waarmee inhoud over meerdere kolommen kan lopen: [CSS Multi-column layout](#). Daarbij kunnen een vast aantal kolommen en de gewenste breedte van de kolommen worden ingesteld. Er worden zo veel kolommen gemaakt als er in de container passen.

De inhoud van de kolommen kan van alles zijn: tekst, afbeeldingen, lijsten, containers met daarin weer tekst en afbeeldingen enzovoort. Binnen de kolommen wordt de normale flow van elementen gevolgd. Multi-column is niet geschikt om complete pagina's op te maken. Het is wel geschikt

voor het in kolommen weergeven van onderdelen van een pagina, bijvoorbeeld artikelen (met afbeeldingen), een paginafooter met veel informatie of productkaarten. Ook een gerubriceerde lijst van producten of iets dat daarop lijkt kan een goede toepassing zijn.

Afbeelding 10.17 *Er zijn geen angewenste marges meer (10_17.html).*

Kolommen overspannen

Met de eigenschap `column-span` laat u een kop, een afbeelding of een quote of meerdere kolommen overspannen. Het is alles of niets: u kunt niet kiezen hoeveel kolommen worden overspannen. Er zijn dan ook maar twee instellingen beschikbaar:

- `none` geen overspanning, de inhoud blijft binnen de kolom (standaard);
- `all` alle kolommen worden overspannen.



Niet in Firefox

De eigenschap `column-span` werkt (nog) niet in Firefox. Bekijk het voorbeeld in Chrome

De volgende code laat een streamer meerdere kolommen overspannen. (Een streamer is een opvallend tekstfragment uit een artikel. Merk op dat het geen `<blockquote>` is, want dan moet een andere site de bron zijn.)

```
<article>
  <p> ... </p>
  <figure>
    <p> ... </p>
  </figure>
  <p> ... </p>
</article>
/* CSS */
figure {
  column-span: all;
}
```

Er is een CSS-techniek waarmee inhoud over meerdere kolommen kan lopen: [CSS Multi-column layout](#). Daarbij kunnen een vast aantal kolommen en de gewenste

breedte van de kolommen worden ingesteld. Er worden zo veel kolommen gemaakt als er in de container passen.

Multi-column is niet geschikt voor paginaopmaak

De inhoud van de kolommen kan van alles zijn: tekst, afbeeldingen, lijsten, containers met daarin weer tekst en afbeeldingen enzovoort. Binnen de kolommen wordt de normale flow van elementen gehvolgd. Multi-column is niet geschikt om complete pagina's op te maken. Het is wel geschikt voor

het in kolommen weergeven van onderdelen van een pagina, bijvoorbeeld artikelen (met afbeeldingen), een paginafooter met veel informatie of productkaarten. Ook een gerubriceerde lijst van producten of iets dat daarop lijkt kan een goede toepassing zijn.

Afbeelding 10.18 Een streamer overspant alle kolommen, maar breekt de tekstkolom (10_18.html).

De onderbreking van de tekst met column-span heeft wel tot gevolg dat de tekst voor de overspanning ook altijd boven de overspanning blijft. De bovenste kolom rechts is de voortzetting van de bovenste kolom links. Onder de overspanning begint een nieuwe alinea. Bij een tussenkop is dit uiteraard precies wat u wilt, maar bij een streamer is voor de lezer de loop van de tekst nogal verwarring. Het is daarom beter zo'n onderbreking binnen de kolom te houden. In elk geval zolang dit gedrag niet is veranderd. Een kop kan natuurlijk wel prima boven aan het artikel alle kolommen overspannen.

Ruimte tussen de kolommen

Tussen de kolommen wordt altijd ruimte vrijgehouden. Dit doet de eigenschap column-gap. De standaardwaarde is normal en die is gelijk aan 1em. De tussenruimte kan worden ingesteld met een lengte. Bedenk dat de tussenruimte ten koste gaat van de ruimte voor de kolommen zelf.

Er is een CSS-techniek waarmee inhoud over meerdere kolommen kan lopen: [CSS Multi-column layout](#). Daarbij kunnen een vast aantal kolommen en de gewenste breedte van de kolommen worden ingesteld. Er worden zo veel kolommen gemaakt als er in de container passen.

De inhoud van de kolommen kan van alles zijn: tekst, afbeeldingen, lijsten, containers met daarin weer tekst en afbeeldingen enzovoort. Binnen de kolommen

wordt de normale flow van elementen gehvolgd. Multi-column is niet geschikt om complete pagina's op te maken. Het is wel geschikt voor het in kolommen weergeven van onderdelen van een pagina, bijvoorbeeld artikelen (met afbeeldingen), een paginafooter met veel informatie of productkaarten. Ook een gerubriceerde lijst van producten of iets dat daarop lijkt kan een goede toepassing zijn.

Afbeelding 10.19 Een extra brede ruimte tussen de kolommen (10_19.html).

```
p {
    column-width: 200px;
    column-gap: 2em;
}
```



Percentage als waarde

In de specificatie wordt ook een percentage als mogelijke waarde genoemd, maar met de opmerking dat die kan vervallen. Net als bij elk ander percentage als waarde in CSS is de breedte van de container de referentie: in een container van 700px is een column-gap van 5% gelijk aan 35px.

Lijn tussen de kolommen

Als grafisch effect kan tussen de kolommen een lijn worden geplaatst, waarvan de dikte, de kleur en de stijl instelbaar zijn. De lijn komt altijd in het midden van de tussenruimte te staan. De lijn neemt geen ruimte in. Als de lijn dikker is dan de tussenruimte, overlapt de tekst de lijn.

De eigenschappen zijn:

- `column-rule-width` de breedte van de lijn, als een lengte of een van de sleutelwoorden `thin`, `medium` of `thick` (standaard `medium`);
- `column-rule-color` een kleurwaarde (standaard `currentcolor`; alle kleurwaarden staan in hoofdstuk 12);
- `column-rule-style` een lijnstijl (standaard geen stijl; lijnstijlen zijn te vinden in de paragraaf *Randstijl*);
- `column-rule` één eigenschap to *rule them all*. Een weggelaten waarde krijgt de standaardwaarde van die eigenschap.

De volgende code maakt een lichtgrijze, doorgetrokken kolomlijn met een breedte van 1px:

```
article {
    column-count: 2;
    column-gap: 2em;
    column-rule-color: darkgreen;
    column-rule-style: solid;
    column-rule-width: 3px;
}
```

In de korte notatie wordt dat:

```
column-rule: 3px solid darkgreen;
```

Er is een CSS-techniek waarmee inhoud over meerdere kolommen kan lopen: [CSS Multi-column layout](#). Daarbij kunnen een vast aantal kolommen en de gewenste breedte van de kolommen worden ingesteld. Er worden zo veel kolommen gemaakt als er in de container passen.

De inhoud van de kolommen kan van alles zijn: tekst, afbeeldingen, lijsten, containers met daarin weer tekst en afbeeldingen

enzovoort. Binnen de kolommen wordt de normale flow van elementen gevolgd. Multi-column is niet geschikt om complete pagina's op te maken. Het is wel geschikt voor het in kolommen weergeven van onderdelen van een pagina, bijvoorbeeld artikelen (met afbeeldingen), een paginafooter met veel informatie of productkaarten. Ook een gerubriceerde lijst van producten of iets dat daarop lijkt kan een goede toepassing zijn.

Afbeelding 10.20 Een lijn tussen de kolommen (10_20.html).

Kolommen vullen

Kolommen kunnen op twee manieren worden gevuld: gebalanceerd of niet. Gebalanceerd wil zeggen dat de browser probeert alle kolommen even hoog te maken, maar daarbij wel rekening houdt met de instellingen voor afbreken. Zonder balans loopt de inhoud er gewoon in en kunnen gedeeltelijk gevulde of lege kolommen ontstaan.

Multi-column layout is gebaseerd op het begrip *fragmentatie*. Fragmentatie treedt op als bijvoorbeeld een deel van alinea naar een volgende kolom of pagina loopt. Hoe een alinea wordt verdeeld in fragmenten, wordt onder meer bepaald door de instellingen voor afbreken in kolommen en het vullen van kolommen.

De eigenschap is `column-fill` en de mogelijke waarden zijn:

- `auto` geen kolombalans;
- `balance` (standaard) balanceert de kolommen zo veel mogelijk, maar bij fragmentatie wordt alleen het laatste fragment gebalanceerd;
- `balance-all` balanceert de kolommen zo veel mogelijk, ook fragmenten.

In de praktijk voldoet de standaardinstelling prima. De waarde `balance-all` wordt nog door geen enkele browser ondersteund.

Afbreken in kolommen

U kunt suggesties doen voor het (niet) afbreken van inhoud in kolommen. Om het politiek te formuleren: de eigenschappen voor het afbreken in kolommen worden door de browser meegenomen bij het plaatsen van de inhoud, maar de browser beslist.



Afbreken in CSS

Alle informatie over afbreken (en bijeenhouden) staat in de module CSS *Fragmentation level 3* (www.w3.org/TR/css-break-3/). Fragmentatie is wat er gebeurt als inhoud wordt afgebroken tussen twee pagina's (op papier) of tussen kolommen of regio's (op een beeldscherm). Deze module geeft hulpmiddelen om dat afbreken enigszins te controleren.

Er zijn drie eigenschappen voor het afbreken in kolommen, in te stellen op de elementboxen of de container. (In de getoonde voorbeelden zijn `<article>` en `<figure>` containers en maken de elementen `<p>` de elementboxen.)

- `break-after` beïnvloedt het afbreken van de volgende box;
- `break-before` beïnvloedt het afbreken van de vorige box;
- `break-inside` beïnvloedt het afbreken in de container.

Voor elk van deze eigenschappen zijn twee waarden speciaal voor meerkolomslay-out beschikbaar:

- `avoid-column` voorkomt een kolomafbreking voor (before) of na (after) de box waarop de eigenschap is ingesteld;
- `column` forceert een kolomafbreking voor of na de box waarop de eigenschap is ingesteld.

Er zijn ook twee algemene waarden die in elke context gelden:

- `auto` afbreken is niet expliciet verboden of toegestaan;
- `avoid` werkt net als `avoid-column`.

In de volgende afbeelding is het de bedoeling dat elke kop boven aan een nieuwe kolom begint. Dit is de code:

```
<article>
  <h2> ... </h2>
  <p> ... </p>
  <h2> ... </h2>
  <p> ... </p>
</article>
/* CSS */
article {
  column-width: 200px;
}
h2 {
  break-before: column;
}
```

Multi-column

Er is een CSS-techniek waarmee inhoud over meerdere kolommen kan lopen: [CSS Multi-column layout](#). Daarbij kunnen een vast aantal kolommen en de gewenste breedte van de kolommen worden ingesteld. Er worden zo veel kolommen gemaakt als er in de container passen.

Inhoud

De inhoud van de kolommen kan van alles zijn: tekst, afbeeldingen,

lijsten, containers met daarin weer tekst en afbeeldingen enzovoort. Binnen de kolommen wordt de normale flow van elementen gevolgd. Multi-column is niet geschikt om complete pagina's op te maken. Het is wel geschikt voor het in kolommen weergeven van onderdelen van een pagina, bijvoorbeeld artikelen (met afbeeldingen), een paginafooter met veel informatie of productkaarten. Ook een gerubriceerde lijst van producten of iets dat daarop lijkt kan een goede toepassing zijn.

Afbeelding 10.21 *Dit gebeurt er zonder breaks.*

Multi-column

Er is een CSS-techniek waarmee inhoud over meerdere kolommen kan lopen: [CSS Multi-column layout](#). Daarbij kunnen een vast aantal kolommen en de gewenste breedte van de kolommen worden ingesteld. Er worden zo veel kolommen gemaakt als er in de container passen.

Inhoud

De inhoud van de kolommen kan van alles zijn: tekst, afbeeldingen, lijsten, containers met daarin weer tekst en afbeeldingen enzovoort. Binnen de kolommen wordt de normale flow van elementen gevolgd. Multi-column is niet geschikt om complete pagina's op te maken.

Het is wel geschikt voor het in kolommen weergeven van onderdelen van een pagina, bijvoorbeeld artikelen (met afbeeldingen), een paginafooter met veel informatie of productkaarten. Ook een gerubriceerde lijst van producten of iets dat daarop lijkt kan een goede toepassing zijn.

Afbeelding 10.22 *Met break-before: column ingesteld op <h2> komt elke kop boven aan de kolom. Doordat de hoeveelheid tekst erg verschilt, heeft het algoritme moeite om er wat van te maken (10_21.html).*

Aan de tekst wordt een afbeelding met bijschrift toegevoegd. Het is de bedoeling dat die niet los van elkaar in verschillende kolommen komen te staan. Daarom wordt aan de container `<figure>` de eigenschap `break-inside: avoid-column` toegevoegd.

```
<article>
  <p> ... </p>
  <figure>
    
    <figcaption> ... </figcaption>
  </figure>
  <p> ... </p>
</article>
```

Multi-column

Er is een CSS-techniek waarmee inhoud over meerdere kolommen kan lopen: [CSS Multi-column layout](#). Daarbij kunnen een vast aantal kolommen en de gewenste breedte van de kolommen worden ingesteld. Er worden zo veel kolommen gemaakt als er in de container passen.



Een meditatief moment

De inhoud van de kolommen kan van alles zijn: tekst, afbeeldingen, lijsten, containers met daarin weer tekst en afbeeldingen enzovoort. Binnen de kolommen wordt de normale flow van elementen gevolgd. Multi-column is niet geschikt om complete pagina's op te maken. Het is wel geschikt voor het in kolommen weergeven van onderdelen van een pagina, bijvoorbeeld artikelen (met afbeeldingen), een

Afbeelding 10.23 Het bijschrift loopt over naar de volgende kolom, dat is niet de bedoeling.

Multi-column

Er is een CSS-techniek waarmee inhoud over meerdere kolommen kan lopen: [CSS Multi-column layout](#). Daarbij kunnen een vast aantal kolommen en de gewenste breedte van de kolommen worden ingesteld. Er worden zo veel kolommen gemaakt als er in de container passen.



Een meditatief moment

De inhoud van de kolommen kan van alles zijn: tekst, afbeeldingen, lijsten, containers met daarin weer tekst en afbeeldingen enzovoort. Binnen de kolommen wordt de normale flow van elementen gevolgd. Multi-column is niet geschikt om complete pagina's op te maken. Het is wel geschikt voor

Afbeelding 10.24 Afbeelding en bijschrift blijven zonder afbrekingen bij elkaar dankzij `break-inside: avoid-column` ([10_23.html](#)).

Wanneer multi-column (niet) nuttig is

Multi-column is lang niet in elke situatie een goede oplossing. Door het krantenkolomeffect lijkt het heel geschikt voor grote artikelen met tussenkoppen, afbeeldingen en streamers, maar hiervoor was te zien dat de tekstloop na `column-span: all` niet logisch is. Een ander probleem ontstaat bij inhoud die langer is dan het scherm hoog. De lezer moet dan eerst omlaag scrollen naar

Multi-column

Er is een CSS-techniek waarmee inhoud over meerdere kolommen kan lopen: [CSS Multi-column layout](#). Daarbij kunnen een vast aantal kolommen en de gewenste breedte van de kolommen worden ingesteld. Er worden zo veel kolommen gemaakt als er in de container passen.



Een meditatief moment

De inhoud van de kolommen kan van alles zijn: tekst, afbeeldingen, lijsten, containers met daarin weer tekst en afbeeldingen enzovoort. Binnen de kolommen wordt de normale flow van elementen gevolgd. Multi-column is niet geschikt om complete pagina's op te maken. Het is wel geschikt voor het in kolommen weergeven van onderdelen van een pagina, bijvoorbeeld artikelen (met afbeeldingen), een

Afbeelding 10.23 Het bijschrift loopt over naar de volgende kolom, dat is niet de bedoeling.

Multi-column

Er is een CSS-techniek waarmee inhoud over meerdere kolommen kan lopen: [CSS Multi-column layout](#). Daarbij kunnen een vast aantal kolommen en de gewenste breedte van de kolommen worden ingesteld. Er worden zo veel kolommen gemaakt als er in de container passen.



Een meditatief moment

De inhoud van de kolommen kan van alles zijn: tekst, afbeeldingen, lijsten, containers met daarin weer tekst en afbeeldingen enzovoort. Binnen de kolommen wordt de normale flow van elementen gevolgd. Multi-column is niet geschikt om complete pagina's op te maken. Het is wel geschikt voor

Afbeelding 10.24 Afbeelding en bijschrift blijven zonder afbrekingen bij elkaar dankzij `break-inside: avoid-column` ([10_23.html](#)).

Wanneer multi-column (niet) nuttig is

Multi-column is lang niet in elke situatie een goede oplossing. Door het kranenkolomeffect lijkt het heel geschikt voor grote artikelen met tussenkoppen, afbeeldingen en streamers, maar hiervoor was te zien dat de tekstloop na `column-span: all` niet logisch is. Een ander probleem ontstaat bij inhoud die langer is dan het scherm hoog. De lezer moet dan eerst omlaag scrollen naar

het eind van de kolom en daarna omhoog scrollen naar het begin van de volgende kolom. Een derde probleem is zijwaarts scrollen, dat kan voorkomen wanneer er meer kolommen worden gemaakt dan het scherm of de container breed is.

Multi-column kan vooral handig zijn als de inhoud beperkt is en de beschikbare ruimte voorspelbaar. Dat er uitzonderingen op die regel zijn bewijst de portfoliotoepassing hierna.

Toepassing: bol.com

Wat is bij zo veel tekortkomingen dan nog een nuttige toepassing voor multi-column? Een goed voorbeeld is een ook op andere sites veelgebruikte menustructuur zoals op bol.com.

Categorieën ▾	Categorieën & voorzetter ▾	Aanbevolgen ▾	Verkoop	Gespot	Overzicht	☰
Boeken	Categorieën	Taal	Populair & nieuw			X
Muziek, Film & Games	Literatuur & Romantiek	Nederlandstalige boeken	Bestsellers gratis downloaden			
	Thrillers & Spanning	Engelse talige boeken	Nieuw verschijnen			
Computer & Elektronica	Kinderboeken	Duitstalige boeken	Te reserveren			
	Studieboeken	Fransstalige boeken				
Speelgoed & Hobby	Gezonchheid & Psychologie	Alle talen	Tips			
Sport & Kind	Kookboeken		Cadeauboeken			
Mooi & Gesond	Reisboeken	Ebooks & E-readers	Tweedehands boeken			
Mode, Schoenen & Accessoires	Fantasy	Kobo Plus abonnement	Virkoop je boeken			
	Kunst & Fotografie	Kobo Plus ebooks	Zelf je boek uitgeven			
Geschiedenis & Maatschappij	Geschiedenis	E-books	Lees Magazine			
Sport, Outdoor & Reizen	Hobby, Huis & Tuin	E-readers				
Kantoor & School	sportboeken	De voorzetter van elke sport	outlet			
Drank	Managementboeken					
Wonen, Koken & Huishouden	Alle boeken					
Dier, Tuin & Kassen						

Afbeelding 10.25 Een deel van het menu Categorieën op bol.com.

Het originele menu op bol.com is gemaakt met flexbox, maar met multi-column kan het zeker ook. De vereenvoudigde HTML-code van de make-over is als volgt (complete code in 10_26.html):

```
<nav>
<ul>
<li>
<ul>
<li><strong>Categorieën</strong></li>
```

```

<li>Literatuur & Romans</li>
  ...
</ul>
</li>
<li class="break">
  <ul>
    <li><strong>Taal</strong></li>
    <li>Nederlandstalige boeken</li>
    ...
  </ul>
</li>
<li class="break">
  <ul>
    <li><strong>Populair & Nieuw</strong></li>
    <li>Bestsellers gratis verzonden</li>
    ...
  </ul>
</li>
</ul>
</nav>
```

De bijbehorende CSS:

```

nav {
  columns: 3;
  padding: 0 1em;
  width: 670px;
}

ul {
  margin-top: 1em;
  padding: 0;
}

li {
  list-style: none;
}

.break {
  break-before: column;
}
```

Analyse van de code

Het menu is alleen bedoeld voor schermen vanaf 900px en heeft een vaste breedte. De inhoud is (min of meer) onveranderlijk en in combinatie met de vaste breedte werkt multi-column hier prima. Met `<nav>` wordt een container gemaakt en hierop wordt het gewenste aantal kolommen ingesteld. Een bu-

Afbeelding 10.26 Het resultaat van de code (10_26.html).

tenste lijst omvat alle hoofdgroepen en daarin staan sublijsten voor de categorieën. Bij de hoofdgroepen die boven aan een kolom moeten staan is break-before: column ingesteld. De vette tekst van de hoofdgroepen komt van .

Toepassing: portfolio

Voor een uitgebreid portfolio is een lay-out met kaarten handig. Zo'n kaart bevat bijvoorbeeld een titel, een afbeelding en een korte toelichting en kan linken naar meer informatie over dat project. De hoogte van de kaarten is afhankelijk van de afbeelding en de hoeveelheid tekst. Daardoor zullen ze niet allemaal even hoog zijn. Met multi-column sluiten toch alle kaarten op elkaar aan.

De ingekorte HTML-code (complete code in 10_27.html):

```
<div class="pf-container">
  <figure class="pf-card">
    <h3>Project</h3>
    
    <p> ... </p>
  </figure>
  ...
</div>
```

De CSS:

```
.pf-container {  
    columns: 200px 4;
```



Afbeelding 10.27 Kaarten van verschillend formaat sluiten verticaal op elkaar aan (10_27.html)

```

column-gap: 1em;
border: 1px solid black;
padding: 1em;

}

.pf-card {
break-inside: avoid;
background-color: #eee;
margin: 0 0 1em 0;
padding: 10px;
}

.pf-card h3 {
margin-top: 0;
}

img {
width: 100%;
}

```

Analyse van de code

Een container omvat alle kaarten. Er zijn maximaal vier kolommen die minimaal 200px breed zijn: `columns: 200px 4`. Er is een tussenruimte van 1em. Op de kaarten (`<figure>`) is ingesteld dat de inhoud niet mag worden afgebroken: `break-inside: avoid`. De kop `<h3>` heeft een bovenmarge van nul om verspringen boven aan de kolom te voorkomen.

Het leuke van deze uitvoering is dat het portfolio bij elke schermbreedte goed wordt weergegeven. Op een smaller scherm staan er gewoon minder kaarten naast elkaar, die op zijn breedst ruim 400px zijn. De verhouding van de afbeeldingen blijft goed, doordat de breedte ervan 100% is. Zo passen ze zich automatisch aan de container aan.

Lay-out in een raster: gridlay-out

Gridlay-out maakt een tweedimensionaal raster. Daar kan een lay-out mee worden gemaakt door paginaonderdelen te koppelen aan rijen en kolommen. Items kunnen meerdere rijen en kolommen overspannen. Grid-lay-out maakt het gedoe met floats overbodig, geeft vergaande controle over de lay-out bij alle schermbreedten en maakt uitlijnen in alle richtingen mogelijk. En dit is nog maar de kortst mogelijke samenvatting van wat er kan met gridlay-out, kortweg grid of, zoals de module officieel heet, *CSS Grid Layout level 1* (www.w3.org/TR/css-grid-1/). Voor het maken van lay-outs met CSS is het een enorme stap vooruit.

In korte tijd zijn aan CSS twee lay-outsystemen toegevoegd: flexbox en gridlay-out. Flexbox wordt hierna besproken, maar het is wel handig om de twee alvast globaal te vergelijken. Een belangrijk verschil is dat flexbox draait om het *verdelen van inhoud in één richting*, terwijl grid *inhoud op een raster plaatst en in twee richtingen tegelijk werkt*. Grid is ontwikkeld voor complete paginalay-outs, al kunnen er ook prima componenten mee worden gebouwd. Flexbox is vooral handig voor componenten die mogen groeien in krimpen in de beschikbare ruimte. Overeenkomsten zijn er zeker ook, zoals de flexibiliteit, de uitlijning in alle richtingen en de mogelijkheid om de weergavevolgorde te laten afwijken van de volgorde van de HTML-bron. De technieken kunnen overigens heel goed samen worden gebruikt, wat in diverse voorbeelden ook gebeurt.

Een logische vraag is wanneer flexbox de beste keus is en wanneer gridlay-out. Daar kunt u op twee manieren naar kijken. Als er maar in één richting controle over de lay-out nodig is, rij óf kolom, gebruik dan flexbox. Is controle nodig over de rijen én de kolommen, gebruik dan gridlay-out. De tweede manier om ernaar te kijken is: bij flexbox is de inhoud het uitgangspunt en bij grid de lay-out. Bij flexbox wordt inhoud verdeeld in een container. Items kunnen groeien

en krimpen om de container te vullen. Bij grid wordt eerst het raster gemaakt en daar wordt de inhoud in geplaatst. Het raster kan overigens wel flexibel zijn.

Belangrijk is ook dat intussen (2019) alle relevante browsers op desktop en mobiel gridlay-out volledig ondersteunen. Een terugvalmechanisme zou alleen nog nodig zijn voor bijvoorbeeld iPhones met iOS 10.2 en ouder en Internet Explorer 10/11.



Flexbox als paginaraster

Flexbox werd eerder door browsers ondersteund dan gridlay-out. Daardoor is in frameworks voor user interfaces (UI-frameworks) zoals Bootstrap en Bulma flexbox de basis van het rastersysteem dat die frameworks bieden, hoewel het daar veel minder geschikt voor is dan gridlay-out. Overigens kunt u heel goed onderdelen van zo'n framework gebruiken en toch zelf een lay-out met grid maken.

Kenmerken van gridlay-out

Gridlay-out heeft een paar belangrijke principes en kenmerken.

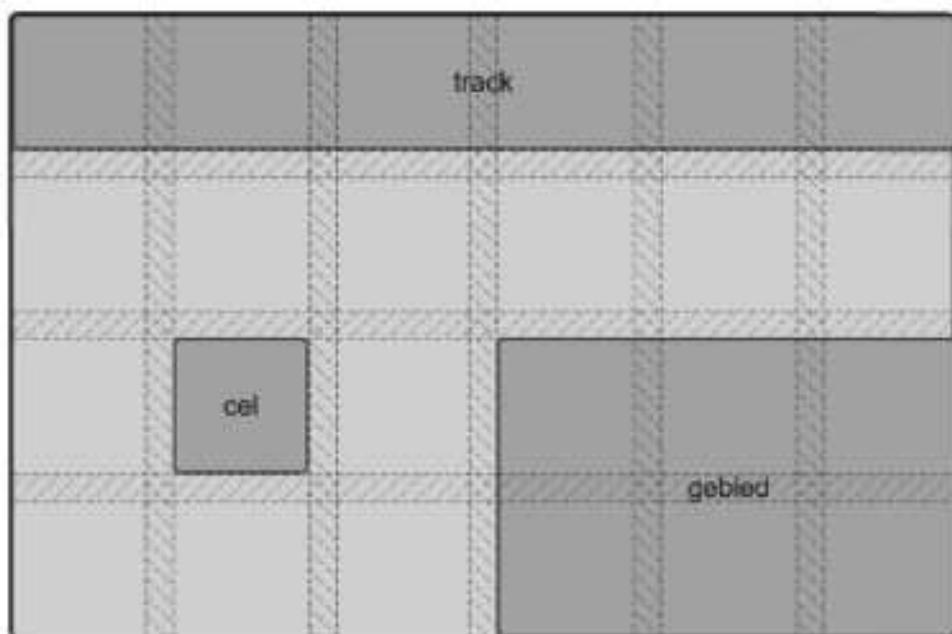
- Met kruisende lijnen wordt in een container een raster gemaakt. Horizontale lijnen maken rijen en verticale lijnen maken kolommen. De rijen en kolommen worden tracks genoemd; die bevatten de rasteritems.
- Tracks kunnen een vaste afmeting hebben in pixels of em, of een flexibele afmeting in procenten of de nieuwe eenheid fr.
- De positie van items kan precies worden bepaald op basis van lijnnummers, lijnen met een naam of gebieden met een naam. Items kunnen ook automatisch worden geplaatst en het is mogelijk om open plekken in het raster te laten opvullen.
- Als items buiten het gedefinieerde raster vallen, worden automatisch de benodigde rijen en kolommen toegevoegd. Het raster kan dus naar behoeft groeien.
- Items kunnen in alle richtingen worden uitgelijnd en tracks kunnen tussenruimte hebben.
- Items kunnen elkaar overlappen en de stapelvolgorde kan worden geregeld met z-index.
- Het is mogelijk om rastercontainers te nesten en zo een raster in een raster te maken.
- Gridlay-out kan worden gecombineerd met flexbox.



Het raster geldt alleen voor kindelementen

Alleen kinderen van een rastercontainer kunnen rasteritems zijn. Als de lay-out niet klopt, controleer dan of u werkt met de juiste elementen.

Uit de kenmerken komen een paar kernbegrippen naar voren. Zo is er de *rastercontainer* (*grid container*) waarin het raster wordt gedefinieerd. De *rasteritems* (*grid items*) zijn de HTML-elementen die op het raster worden geplaatst. Alleen kinderen van een rastercontainer kunnen rasteritems zijn. Het raster wordt opgebouwd met horizontale en verticale *rasterlijnen* (*grid lines*). De ruimte tussen twee rasterlijnen wordt een *track* genoemd, dat kan een *rij* of een *kolom* zijn. Het kleirste deel van het raster is een *rastercel*. Het vlak waarin een rasteritem wordt geplaatst is een *rastergebied* (*grid area*) en dat kan een of meer cellen omvatten. Tracks kunnen zijn gescheiden door *tussenruimte* (*gap*).



Afbeelding 10.28 Kenmerkende onderdelen van gridlay-out. De arcering is tussenruimte.

Hulpmiddelen voor gridlay-out en flexbox

Er is een goede reden om bij het werken met gridlay-out en flexbox de ontwikkelaarshulpmiddelen van Firefox te gebruiken. Daarin zijn namelijk de CSS Grid Inspector en de CSS Flexbox Inspector opgenomen. Hiermee worden de rijen, kolommen, lijnen en tussenruimten van een raster en de container en vrije ruimte van een flexcontainer uitgelicht. Ook lijnnummers en namen van rastervlakken worden getoond. Dat maakt het opbouwen en analyseren van raster- en flexcontainers een stuk eenvoudiger.

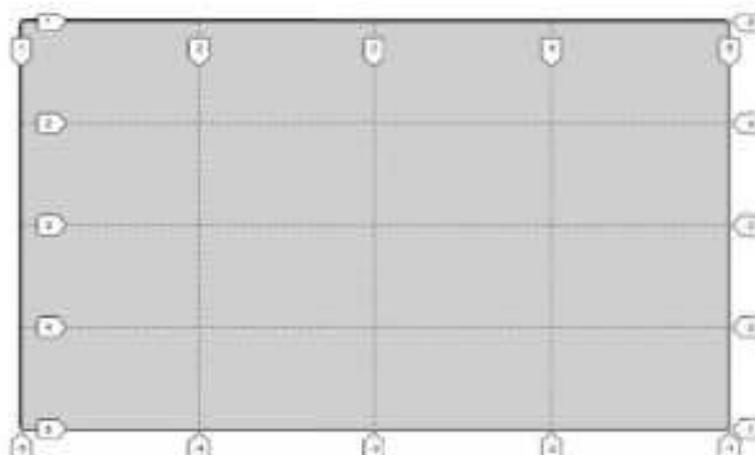
In Chrome kan wel het raster zichtbaar worden gemaakt door in de HTML-weergave een rastercontainer of een rasteritem aan te wijzen. Lijnnummers of namen van gebieden worden niet weergegeven en zodra de focus verschuift, verdwijnt de rasteroverlay. Voor Chrome zijn er wel extensies die meer kunnen, bijvoorbeeld Gridman.

Open in Firefox met Ctrl+Shift+I de Inspector. Als grid of flexbox op de pagina is toegepast, is in de HTML-weergave bij het element een label met de naam grid of flex te zien. In de CSS-weergave staat er dan een pictogram dat een raster of een flexcontainer voorstelt. Klik op het label of het pictogram om het raster of het flexobject zichtbaar te maken. Klik in het CSS-deel van de hulpmiddelen op het tabblad Indeling voor de opties van de Grid Inspector.

```
<!DOCTYPE html>
<html lang="nl"> event
  > <head> </head>
  > <body>
    > <div class="grid"> </div>
  </div>
```

.grid {
 display: grid;
 grid-template-columns: repeat(4, 1fr);
 grid-template-rows: repeat(4, 1fr);
}

Afbeelding 10.29 Een label in de HTML-weergave en een pictogram bij de CSS-regels duiden aan dat gridlay-out is.



Afbeelding 10.30 Het grid is zichtbaar gemaakt.



Afbeelding 10.31 De instellingen voor de Grid Inspector van Firefox.

Open in Firefox met Ctrl+Shift+I de Inspector. Als grid of flexbox op de pagina is toegepast, is in de HTML-weergave bij het element een label met de naam grid of flex te zien. In de CSS-weergave staat er dan een pictogram dat een raster of een flexcontainer voorstelt. Klik op het label of het pictogram om het raster of het flexobject zichtbaar te maken. Klik in het CSS-deel van de hulpmiddelen op het tabblad Indeling voor de opties van de Grid Inspector.

```
<!DOCTYPE html>
<html lang="nl"> event
  > <head> 4</head>
  > <body>
    > <div class="grid"> 5</div>
  </div>
```

.grid {
 display: grid;
 grid-template-columns: repeat(4, 1fr);
 grid-template-rows: repeat(4, 100px);
}

Afbeelding 10.29 Een label in de HTML-weergave en een pictogram bij de CSS-regels duiden aan dat gridlay-out is.



Afbeelding 10.30 Het grid is zichtbaar gemaakt.



Afbeelding 10.31 De instellingen voor de Grid Inspector van Firefox.

Een raster maken

Een raster wordt op blokniveau gemaakt door op een container de eigenschap `display` de waarde `grid` te geven. De waarde kan ook `inline-grid` zijn, dan wordt een raster op regelniveau gemaakt.

We beginnen met de volgende minimale HTML-code:

```
<div class="grid">  
</div>
```

Vervolgens wordt met CSS een raster ingesteld:

```
.grid {  
    display: grid;  
}
```

Met deze declaratie wordt het element met de klasse `.grid` een rastercontainer en alle kindelementen zijn rasteritems. Er zijn op dit moment nog geen rijen en kolommen. De volgende CSS-regels maken een raster van vier kolommen en vier rijen in het element met de klasse `.grid`.

```
.grid {  
    display: grid;  
    grid-template-columns: 150px 150px 150px 150px;  
    grid-template-rows: 100px 100px 100px 100px;  
}
```

Nu is in Firefox het raster te zien, zelfs zonder dat er items in geplaatst zijn. Elke waarde bij `grid-template-columns` of `grid-template-rows` staat voor één track. Een track is één rij of één kolom. Hier zijn alle kolommen 150px breed en de rijen zijn 100px hoog. Dit raster is niet flexibel, want de tracks hebben vaste afmetingen. Als er meer inhoud is dan in de cel past, ontstaat overloop.



Afbeelding 10.32 Een raster zonder inhoud (10_32.html).

De functie repeat()

Voor meer rijen of kolommen met dezelfde afmeting is er een werkbesparende functie: `repeat()`. Hiermee kan worden aangeven hoeveel tracks van dezelfde afmeting er moeten zijn. Tussen het aantal en de afmeting moet een komma staan. Het voorbeeld kan ook worden geschreven als:

```
grid-template-columns: repeat(4, 150px);
grid-template-rows: repeat(4, 100px);
```



Het raster laten bouwen met repeat()

Het is mogelijk om met `repeat()` en het sleutelwoord `auto-fill` of `auto-fit` het raster automatisch te laten opbouwen. Dit wordt verderop uitgelegd.

De eenheid fr

Met grid is een nieuwe eenheid ingevoerd: de lengte `fr`. `1fr` staat voor een deel (*fraction*) van de beschikbare ruimte in de rastercontainer. Gelijke waarden van `fr` geven gelijke afmetingen. Een voorbeeld: met drie kolommen van `2fr`, `4fr` en `1fr` wordt de beschikbare ruimte verdeeld over $2 + 4 + 1 = 7$ delen. Bij een container van `1050px` staat `1fr` dan voor `150px`. De kolommen in dit voorbeeld zijn dan `300px`, `600px` en `150px`.

Bij `fr` draait alles om *beschikbare ruimte*. Van de totale ruimte (de breedte van de container of een venster) worden eerst alle vast ingestelde breedtes van kolommen afgetrokken. Wat overblijft is de beschikbare ruimte. Deze ruimte wordt verdeeld naar het aantal eenheden `fr`. Een voorbeeld:

```
.grid {
  display: grid;
  grid-template-columns: 150px 1fr 2fr 150px;
  grid-template-rows: repeat(4, 100px);
}
```

Er zijn vier kolommen, waarvan er twee een vaste breedte van `150px` hebben. Van de totale breedte wordt dus `300px` afgetrokken en de rest wordt verdeeld over `3fr`. Bij een container van `900px` blijft `600px` over. Dan is `1fr` gelijk aan $600 / 3 = 200px$.

In de praktijk wordt de `fr` gebruikt om een flexibel raster te maken is het niet nodig om te weten hoe lang een `fr` is. De inhoud wordt gewoon proportioneel verdeeld en vaak is de grootte van de viewport de enige beperking. Het is wel handig om een minimale afmeting (`min-width` of `min-content`) in te stellen of



Afbeelding 10.33 Het resultaat van dit voorbeeld.

om met media queries de kolomindeling aan te passen als kolommen te smal worden. (Zie de paragraaf *Responsive design*).

Rasteritems plaatsen op lijnnummers

Het raster is gemaakt. Nu worden aan de rastercontainer vier kindelementen toegevoegd: de rasteritems.

```
<div class="grid">
  <div class="grid-item">item 1</div>
  <div class="grid-item">item 2</div>
  <div class="grid-item">item 3</div>
  <div class="grid-item">item 4</div>
</div>
```

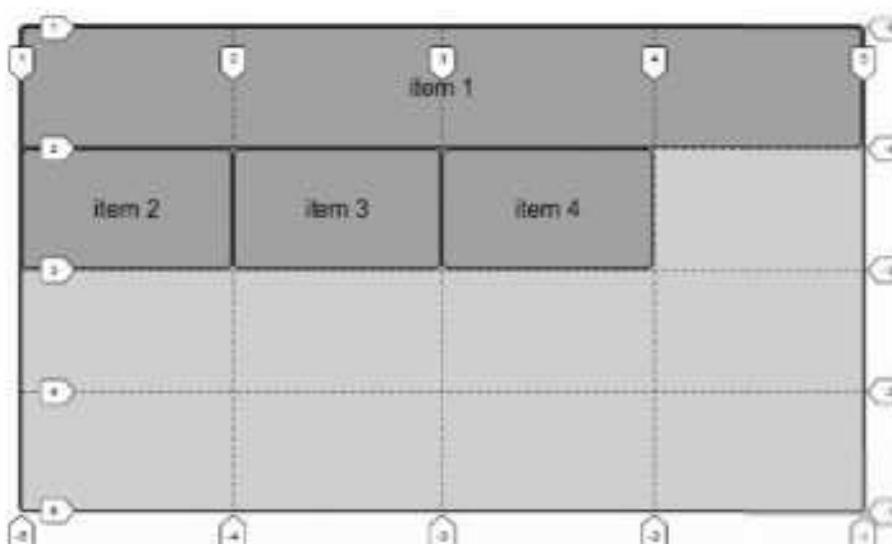
Als niet wordt aangegeven waar de items in het raster moeten staan, worden ze in de volgorde van de HTML in de rijrichting geplaatst, elk in een cel van het raster. In dit voorbeeld is de eerste rij van vier cellen gevuld, omdat er vier kindelementen/rasteritems zijn.

Nu wordt item 1 geplaatst op rij 1, kolom 1 tot en met 4:

```
.grid-item:nth-child(1) {
  grid-column-start: 1;
  grid-column-end: 5;
  grid-row-start: 1;
  grid-row-end: 2;
}
```



Afbeelding 10.34 De vier items zijn in het raster geplaatst (10_34.html)



Afbeelding 10.35 Item 1 staat op zijn plaats (10_35.html).

Het item is geplaatst op basis van de genummerde lijnen. In Firefox is goed te zien dat de eerste kolom begint bij lijn 1 en de vierde kolom eindigt bij lijn 5 (er is logischerwijs altijd een lijn meer dan er kolommen zijn). De eerste rij begint bij lijn 1 en eindigt bij lijn 2.

De declaratie kan korter door begin en eind te combineren in één eigenschap, namelijk `grid-column` en `grid-row`. De twee waarden worden gescheiden door een slash en het is goed gebruik om spaties rond de slash te plaatsen. De volgende regels doen precies hetzelfde:

```
.grid-item:nth-child(1) {  
    grid-column: 1 / 5;
```

```
grid-row: 1 / 2;
}
```

In plaats van het lijnnummer waarop het item eindigt, kan de overspanning worden aangegeven. Dit item overspant vier kolommen en dan wordt de declaratie:

```
.grid-item:nth-child(1) {
  grid-column: 1 / span 4;
  grid-row: 1 / 2;
}
```

Deze declaratie kan nog wat korter. Standaard wordt één track overspannen. Bij een item dat maar één track overspant, kan de eindlijn worden weggelaten.

```
.grid-item:nth-child(1) {
  grid-column: 1 / span 4;
  grid-row: 1;
}
```



Het kan nog korter, maar ...

Als het startpunt wordt weggelaten, wordt dat automatisch de plaats die het item in de flow heeft; bij item 1 is dat rij 1 en kolom 1. Het gevaar van deze werkwijze is dat de plaats in de flow kan veranderen door de plaatsing van andere items. Rasteritems kunnen elkaar overlappen, maar ook wegduwen als de posities niet 'hard' zijn ingesteld. Het is daarom verstandig om altijd het startpunt aan te geven.



Het eind is altijd -1

In plaats van te zoeken naar het hoogste lijnnummer kan de laatste rij- of kolomlijn (het eind van het raster dus) ook worden aangeduid met -1. In de CSS Grid Inspector van Firefox is te zien dat elke positieve positie een gelijke negatieve positie heeft. De beginlijn in dit voorbeeld is dus 1, maar ook -5.

Nu moeten ook de andere items worden geplaatst. De CSS voor item 2 wordt hier gegeven. Probeer zelf deze code uit te breiden voor de items 3 en 4. Dat hoeft niet per se met de korte notatie te zijn. De oplossing is te vinden in de voorbeeldbestanden.

- item 2: rij 2 en 3, kolom 1 en 2
- item 3: rij 3 en 4, kolom 3 en 4
- item 4: rij 4, kolom 2 tot eind

```
.grid-item:nth-child(2) {
    grid-column: 2 / span 2;
    grid-row: 2 / span 2;
}
```



Afbeelding 10.36 Het eindresultaat van de oefening. De items 3 en 4 overlappen in het gearceerde gebied. Ze bezetten beide de kolommen 3 en 4 op rij 4 (10_36.html).

Kolommen en rijen samen in grid-area

De eigenschap `grid-area` is een korte notatie voor `grid-column` en `grid-row`. Bekijk het volgende voorbeeld:

```
.grid-item:nth-child(1) {
    grid-row-start: 1;
    grid-column-start: 1;
    grid-row-end: 2;
    grid-column-end: 5;
}
```

Dit kan ook worden geschreven als:

```
.grid-item:nth-child(1) {
    grid-area: 1 / 1 / 2 / 5;
}
```

De volgorde van de lijnnummers is: rijstart, kolomstart, rij-eind, kolomeind.

In plaats van nummers kunnen ook lijnnamen worden gebruikt.

De lijnen een naam geven

Het tellen van de lijnen om items te plaatsen is te doen, zeker met het hulp-middel van Firefox. Toch is het niet de duidelijkste manier om de posities aan te geven. Een naam die beschrijft welke positie het is, is handiger, zeker naarmate een raster groter en ingewikkelder is. Toegepast op het voorbeeldraster zou dat er zo uit kunnen zien:

```
.grid {
    display: grid;
    grid-template-columns: [col-start] 1fr [col-1-end] 1fr [col-center] 1fr 1fr [col-end];
    grid-template-rows: [row-start] 100px [row-1-end row-2-start] 100px [row-center] 100px
    [row-4-start] 100px [row-end];
}
```

Een lijennaam in de sjabloon van het raster staat tussen blokhaken. Staat de naam voor een waarde, dan heeft die betrekking op de startlijn van een track, bijvoorbeeld [col-start] en [row-start]. Een naam na een waarde geeft de eindlijn aan, bijvoorbeeld bij [col-end]. Tussen de blokhaken mogen meerdere namen staan, bijvoorbeeld [row-1-end row-2-start]. Ze duiden precies dezelfde positie aan, maar bij het plaatsen van de items is dan duidelijker welke track wordt bedoeld.

Met de volgende CSS worden de rasteritems op precies dezelfde manier geplaatst als hiervoor met de lijnummers. Let op, bij het plaatsen staan de namen niet tussen blokhaken.

```
.grid-item:nth-child(1) {
    grid-column: col-start / span 4;
    grid-row: row-start / row-1-end;
}

.grid-item:nth-child(2) {
    grid-column: col-start / span 2;
    grid-row: row-2-start / span 2;
}

.grid-item:nth-child(3) {
    grid-column: col-center / span 2;
    grid-row: row-center / span 2;
}

.grid-item:nth-child(4) {
    grid-column: col-2-start / col-end;
    grid-row: row-4-start / row-end;
}
```



Afbeelding 10.37 Geen nummers maar namen, het resultaat blijft hetzelfde (10_37.html).

In het voorgaande voorbeeld heeft elke lijn een unieke naam, maar dat hoeft niet. In een rastersysteem dat bijvoorbeeld de hele pagina verdeelt in twaalf gelijke kolommen, kan elke startlijn van een kolom de naam col-start hebben. Welke col-start wordt bedoeld, wordt aangegeven met het nummer van de lijn (toch weer). Dit is een voorbeeld van de rasterdefinite:

```
grid-template-columns: repeat(12, [col-start] 1fr);
grid-template-rows: [header-start] auto [article-start] auto;
```

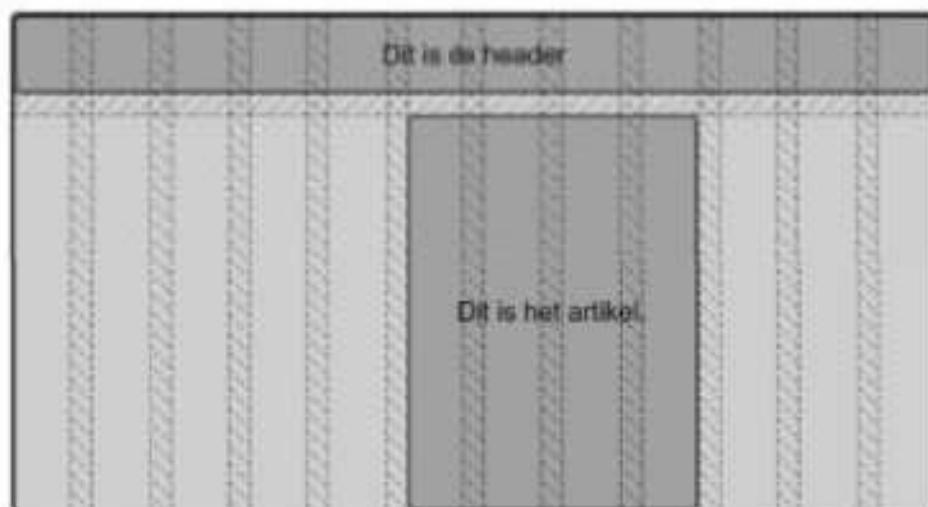
Plaats een rasteritem vanaf kolom 1 over de volle breedte:

```
grid-column: col-start / -1;
grid-row: header-start;
```

Plaats een rasteritem over vier kolommen vanaf kolom zes:

```
grid-column: col-start: 6 / span 4;
grid-row: article-start;
```

Dergelijke rastersystemen lijken erg op opmaakrasters die bij kranten en tijdschriften worden gebruikt. De kracht ervan is dat elke component een vast aantal kolommen krijgt toegewezen, wat een consistente opmaak mogelijk maakt. Omdat elke kolom een flexibele breedte heeft met de eenheid fr, blijft de verdeling altijd in verhouding. Vandaar ook de flexibele eenheid fr.



Afbeelding 10.38 Items in een rastersysteem van twaalf kolommen. De vergroting is tussenruimte (gap) tussen de tracks (10_38.html).

Items plaatsen in gebieden met een naam

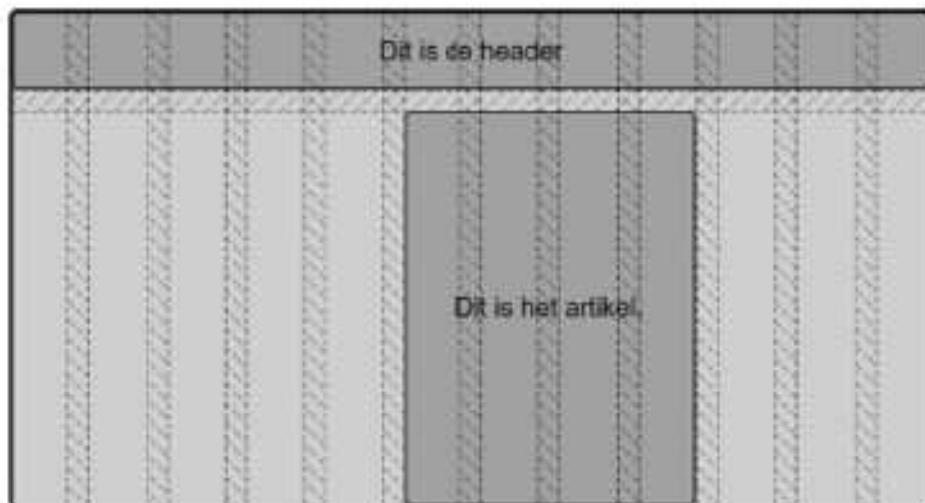
Het is mogelijk om gebieden te markeren met een naam en een item te plaatsen. Op de achtergrond blijven de lijnen het raster bepalen, maar daar hebt u met gemarkeerde gebieden geen omkijken naar. Ook het werken met gebieden begint met een sjabloon voor de kolommen en rijen, maar daarna verzorgt de eigenschap `grid-template-areas` de magie. Met de eigenschap `grid-area` op de rasteritems worden de items geplaatst. (Dit is dezelfde verkorte eigenschap waarmee eerder de rijen en kolommen werden ingesteld.)

Neem een standaardvoorbeeld met een header, hoofdinhoud, zijkolom en footer. De HTML is:

```
<body>
  <div class="grid">
    <header> ... </header>
    <main> ... </main>
    <aside> ... </aside>
    <footer> ... </footer>
  </div>
</body>
```

Geef aan in welk vlak elk rasteritem moet komen te staan. Kies bij voorkeur een korte, duidelijke naam:

```
header {
  grid-area: hd;
}
main {
```



Afbeelding 10.38 Items in een rastersysteem van twaalf kolommen. De arcering is tussenruimte (gap) tussen de tracks (10_38.html).

Items plaatsen in gebieden met een naam

Het is mogelijk om gebieden te markeren met een naam en een item te plaatsen. Op de achtergrond blijven de lijnen het raster bepalen, maar daar hebt u met gemarkeerde gebieden geen omkijken naar. Ook het werken met gebieden begint met een sjabloon voor de kolommen en rijen, maar daarna verzorgt de eigenschap `grid-template-areas` de magie. Met de eigenschap `grid-area` op de rasteritems worden de items geplaatst. (Dit is dezelfde verkorte eigenschap waarmee eerder de rijen en kolommen werden ingesteld.)

Neem een standaardvoorbeeld met een header, hoofdinhoud, zijkolom en footer. De HTML is:

```
<body>
  <div class="grid">
    <header> ... </header>
    <main> ... </main>
    <aside> ... </aside>
    <footer> ... </footer>
  </div>
</body>
```

Geef aan in welk vlak elk rasteritem moet komen te staan. Kies bij voorkeur een korte, duidelijke naam:

```
header {
  grid-area: hd;
}
main {
```

```

grid-area: main;
}

aside {
  grid-area: aside;
}

footer {
  grid-area: ft;
}

```

Definieer een raster van vier kolommen en vier rijen:

```

.grid {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  grid-template-rows: repeat(4, 100px);
}

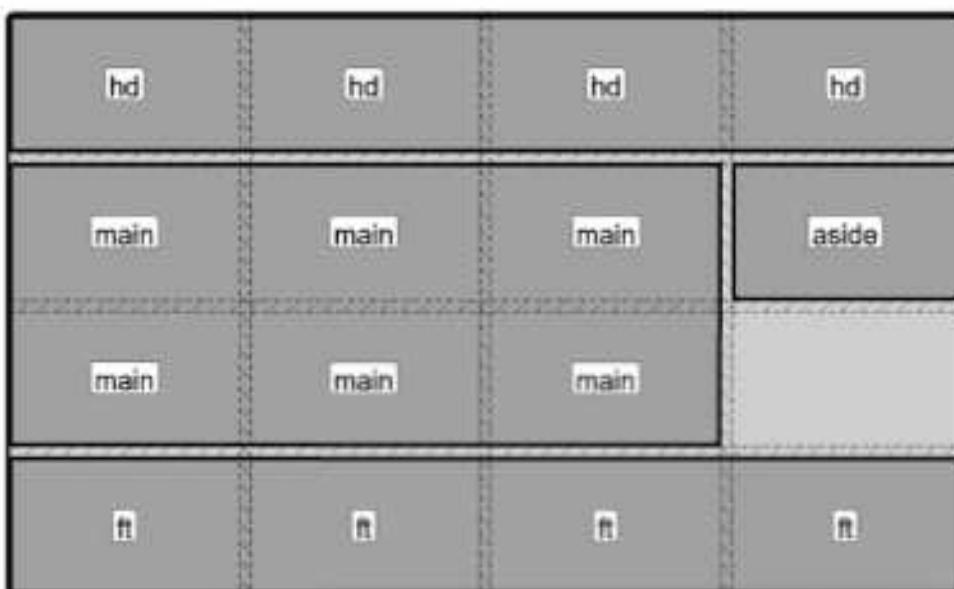
```

Voeg de sjabloon voor de vlakken toe:

```

.grid {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  grid-template-rows: repeat(4, 100px);
  grid-template-areas:
    "hd hd hd hd"
    "main main main aside"
    "main main main ."
    "ft ft ft ft"
}

```



Afbeelding 10.39 De verdeling van de gebieden is te bekijken in Firefox.

Dit is de header			
Dit is het gebied met de belangrijkste inhoud. Hier staan de artikelen.			Hier staat informatie die te maken heeft met de hoofdinhoud.
Dit is de footer van de pagina.			

Afbeelding 10.40 Items geplaatst op basis van gebieden met een naam (10_40.html).

Elke cel in de sjabloon krijgt een naam, die bepaalt u zelf. Een naam staat voor één cel. Om een header met de naam hd vier kolommen breed te maken, wordt de naam hd vier keer herhaald. Elke rij namen staat tussen dubbele aanhalingsstekens. Om een item over meerdere rijen te laten lopen, zet u de naam meerdere keren onder elkaar, zoals bij main. Het is ook mogelijk om een of meer cellen leeg te laten, zet daarvoor op elke lege positie een punt. In dit voorbeeld is dat de cel op regel 3 onder aside.

Afmetingen van tracks instellen

De afmetingen van de tracks , kunnen op verschillende manieren worden ingesteld; enkele voorbeelden kwamen al voorbij. De waarden bij grid-template-rows en grid-template-columns worden een *track-list* genoemd. Deze waarden kunnen zijn:

- namen van lijnen, altijd in combinatie met een lengte of een sleutelwoord;
- lengtewaarden zoals px, % of fr;
- sleutelwoorden.

De mogelijke sleutelwoorden zijn:

- max-content de optimale afmeting van het grootste item in de track, bijvoorbeeld een hele alinea tekst;
- min-content de kleinste mogelijke afmeting zonder overflow te veroorzaken, bijvoorbeeld het langste woord;

- `minmax(min, max)` maakt flexibele tracks met minimaal de afmeting `min` en maximaal de waarde `max`. Min en max kunnen alle voorgaande waarden hebben, dus een lengte of een sleutelwoord zoals `max-content`;
- `auto` als maximum zo groot als kan, als minimum zo groot als nodig;
- `fit-content (waarde)` werkt net als `auto`, maar het formaat van de track wordt beperkt tot de gegeven waarde. Die waarde kan een lengte of een percentage zijn. Het komt erop neer dat alle inhoud wordt geplaatst, maar dat de breedte aan een maximum is gebonden.

Een track-list kan allerlei combinaties van waarden en functies hebben. Een voorbeeld:

```
grid-template-columns: minmax(20px, 1fr) repeat(12, minmax(0, 100px)) minmax(20px, 1fr);
```

Deze track-list maakt een raster van veertien kolommen. Het hart bestaat uit twaalf kolommen van minimaal 0 en maximaal 100px breed. Dat betekent dat op een desktopscherm het hoofdgebied van de pagina maximaal 1200px breed is. (Let op, als tussenruimte wordt ingesteld, komt die er nog bij.) De twee buitenste kolommen zijn minimaal 20px breed en maximaal zo breed als nodig om de ruimte links en rechts van de hoofdinhoud op te vullen. De hoofdinhoud staat zo altijd in het midden van het scherm en er is altijd een marge van 20px tussen inhoud en vensterrand.



Afbeelding 10.41 Veertien flexibele kolommen met de inhoud in de middelste twaalf (10_41.html).

Ruimte tussen tracks

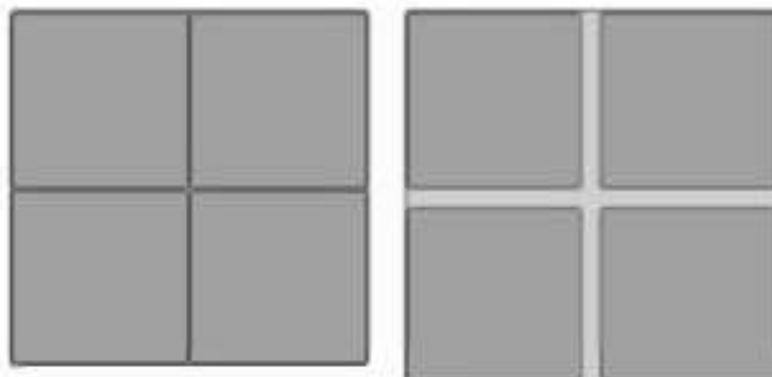
Tussen de tracks kan ruimte worden vrijgehouden. Deze tussenruimte wordt `gap` genoemd. Zie het als een marge op de rijen en kolommen, maar dan alleen binnen het raster. De eigenschap voor ruimte tussen de rijen is `grid-row-gap` en voor kolommen `grid-column-gap`. In de korte notatie `grid-gap` is de eerste waarde voor de rij en de tweede voor de kolom. Als één waarde wordt gegeven, geldt die voor beide. Mogelijke waarden zijn een lengte of een percentage.

Het meer ontwaakt

Lorem ipsum dolor sit amet, consetetur sadipscing elit. Ea massa, nisl nisi. Aliquam doceatur dolor est fugiat magnis in rei omnis quidem repudietur. Accusamus aspernatur esse modi sibi non videat. Recepti. Apud eum interclusa discutitur. Allegorij et euopeum fugit. Itur pueri. Spurium bellicosus maxime traxerat. Non receptare nollet praevaricari. Quod admodum quis recipiens dicitur? Teneatur, unde vides, visusque? Accusamus aspernatur. Conspicillar corpora diversitate non nescia quae representandas velutatae. Visusque?

Bij items met vaste afmetingen wordt tussenruimte opgeteld bij de afmetingen van het raster. Het raster wordt dus groter. Bij een flexibel raster met een breedte van 100vw en tussenruimte blijft het raster wel gewoon binnen de viewport.

De tussenruimte heeft geen eigenschap voor kleur, maar is transparant. De achtergrondkleur van het raster bepaalt dus de kleur van de tussenruimte.



Afbeelding 10.42 Tussenruimte (*gap*) is transparant en wordt bij items met vaste afmetingen opgeteld bij de afmetingen van het raster (10_42.html).

Het raster zichzelf laten bouwen

Tot nu toe zijn steeds alle rijen en kolommen gedefinieerd om de inhoud te kunnen plaatsen. Dat hoeft net per se. Als er meer items zijn dan in de definitie van het raster passen, worden er automatisch rijen en kolommen toegevoegd tot elk onderdeel in het raster staat. De grens voor uitbreiding ligt bij een vaste hoogte of breedte van het raster, of de breedte van het browservenster.



Het expliciete en het impliciete raster

De door de developer in de CSS gedefinieerde tracks vormen het *expliciete raster*. De automatisch toegevoegde tracks worden het *impliciete raster* genoemd.

Zonder instellingen wordt het formaat van de automatisch gemaakte tracks afgestemd op de inhoud, maar het is ook mogelijk om ze afmeting of zelfs verschillende afmetingen te geven. Daarvoor zijn er de eigenschappen `grid-auto-columns` en `grid-auto-rows`. Mogelijke waarden voor `grid-auto-columns` en `grid-auto-rows` zijn hetzelfde als bij `template-rows` en `template-columns`, met twee logische uitzonderingen: lijnen met een naam en de functie `repeat()`.



Firefox kan maar één waarde aan

Firefox heeft een bug waardoor het niet mogelijk is om twee of meer lengtes in te stellen op `grid-auto-columns/rows`. Het is niet bekend wanneer dit wordt opgelost. Het is wel mogelijk om `minmax()` te gebruiken.

Een zelfvoorzienend raster is een goed uitgangspunt voor pagina's met een groot of onbekend aantal items, bijvoorbeeld een Pinterest-achtige pagina of een fotogalerij. Helemaal vanzelf gaat het niet. Er moet wel regelmaat in de afmetingen van de items zitten en het handigst zijn items van dezelfde grootte die steeds één cel gebruiken. (Het kan ook met verschillende afmetingen, maar dan moet op de items worden ingesteld of ze een of meer tracks moeten overspannen. Dat is meer handwerk en minder automatisch.)

Een tweede voorwaarde is dat er een flexibel aantal rijen of kolommen worden gedefinieerd. Dat is een klusje voor de functie `repeat()`. In eerdere voorbeelden hebben we gezien dat `repeat()` een vast aantal tracks kan maken (`repeat(12, 1fr)`), maar in plaats van een vast aantal kan ook het sleutelwoord `auto-fill` of `auto-fit` worden gebruikt. Daarmee wordt het aantal tracks afhankelijk van de beschikbare ruimte. In combinatie met `repeat()` levert dat een volledig flexibel raster op.



Het verschil tussen `auto-fill` en `auto-fit`

De sleutelwoorden `auto-fill` en `auto-fit` maken beide zo veel tracks als er in het raster passen. Het verschil is dat tracks die leeg blijven (doordat er minder items dan tracks zijn) bij `auto-fit` worden ingevouwen. Bij `auto-fill` blijven ze gewoon leeg. Als bij `auto-fit` tracks die items bevatten een flexibele afmeting hebben, bijvoorbeeld `minmax(100px, 1fr)`, kunnen die items groeien om de vrijegekomen ruimte te benutten. Een voorbeeld staat in de paragraaf *Het verschil tussen `auto-fill` en `auto-fit`*.

In dit voorbeeld wordt `auto-fill` gebruikt. Er zijn elf items en voor het raster wordt alleen flexibele kolommen gedefinieerd. De ingekorte HTML is:

```
<div class="grid">
  <div>Item 0</div>
  ...
</div>
```

De CSS is niet veel meer:

```
.grid {  
    display: grid;  
    grid-gap: 10px;  
    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));  
}
```



Afbeelding 10.43 *Op een breed scherm staan alle items naast elkaar (10_43.html).*



Afbeelding 10.44 *Als het scherm smaller wordt, lopen de items door op de volgende regel.*



Afbeelding 10.45 *Op een smal scherm staan alle items onder elkaar.*

Analyse van de code

Het raster heeft geen vaste breedte of hoogte; het browservenster bepaalt de breedte. De kolommen worden gemaakt door de regel:

```
grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
```

- repeat(auto-fit, ...) maakt zo veel kolommen als er passen in het browservenster;
- minmax(200px 1fr) bepaalt dat de kolommen minimaal 200px breed zijn. Als er ruimte overblijft die te klein is voor een nieuwe kolom, gebruiken de andere kolommen een evenredig deel van de beschikbare ruimte om breder te worden (1fr).

Het verschil tussen auto-fill en auto-fit

In dit voorbeeld zijn er zo veel items dat zelfs op een 27-inch scherm de hele breedte wordt benut. Maar wat gebeurt als er maar een paar items zijn? We bekijken een voorbeeld met drie items. In het eerste geval wordt auto-fit gebruikt en in het tweede auto-fill. De rest van de code blijft hetzelfde.



Afbeelding 10.46 Bij auto-fill blijven onbenutte cellen leeg. Bij auto-fit worden lege cellen ingeklappt en wordt de vrijgekomen ruimte benut door de items (10_46.html).

Bij auto-fit wordt telkens als er weer 200px vrije ruimte is een nieuwe kolom gemaakt, ook als er geen item is om die kolom te vullen. Bij auto-fill wordt die nieuwe kolom ook gemaakt, maar als er geen item voor is, wordt de kolom ingevouwen en gebruiken de items de ruimte om te groeien. Dit effect is vergelijkbaar met wat er gebeurt bij flexbox.

Van richting veranderen met grid-auto-flow

De richting waarin items in het raster worden geplaatst, wordt bepaald door de eigenschap grid-auto-flow. De standaardwaarde is row, wat betekent dat items in de rijrichting worden geplaatst tot de regel vol is en dan verdergaan op de volgende regel. De andere waarde is column en dan worden items geplaatst in de blokrichting, gestapeld van boven naar beneden.

Helemaal vanzelf gaat het niet: als de richting row is, worden niet vanzelf (impliciete) kolommen gemaakt. Er is altijd één kolom, dus bij row worden de items gewoon op elkaar gestapeld. Is de richting column, dan worden niet van-

zelf nieuwe rijen gemaakt. Er is altijd één rij en die wordt gewoon eindeloos gevuld en de items lopen desnoods het browservenster uit.

Om dergelijke problemen te voorkomen:

- gebruik bij grid-auto-flow: row altijd grid-template-columns, kan met auto-fill of auto-fit;
- gebruik bij grid-auto-flow: column altijd grid-template-rows, alleen met vaste waarde.

grid-auto-flow: row
grid-template-columns: repeat(auto-fill, minmax(200px, 1fr))

Item 01	Item 02	Item 03	Item 04
Item 05	Item 06	Item 07	Item 08

grid-auto-flow: column
grid-template-rows: repeat(3, auto);

Item 01	Item 04	Item 07	
Item 02	Item 05	Item 08	
Item 03	Item 06		

Afbeelding 10.47 Items plaatsen in de richting rij of kolom (10_47.html).

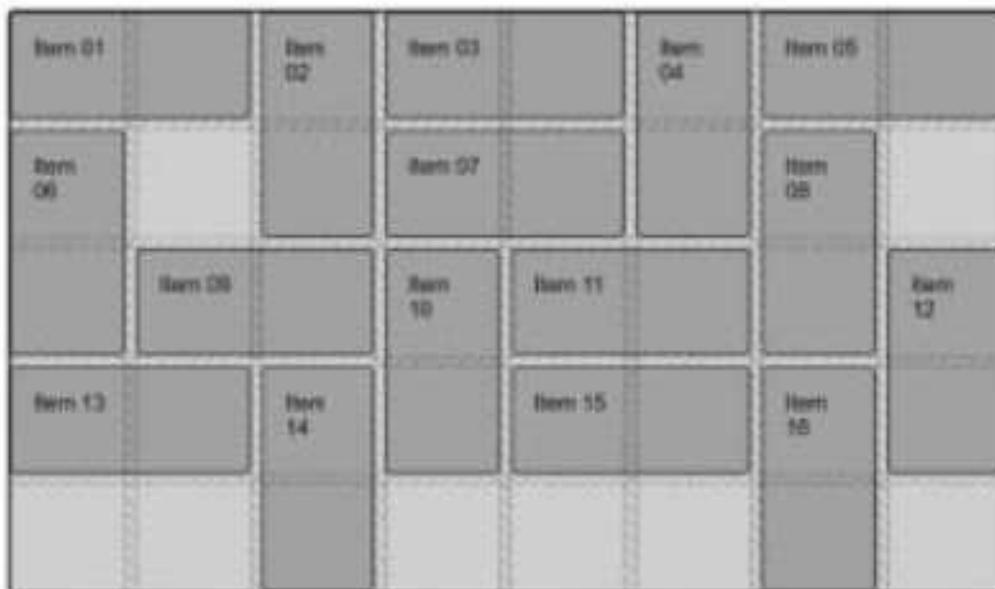


De volgorde veranderen met order

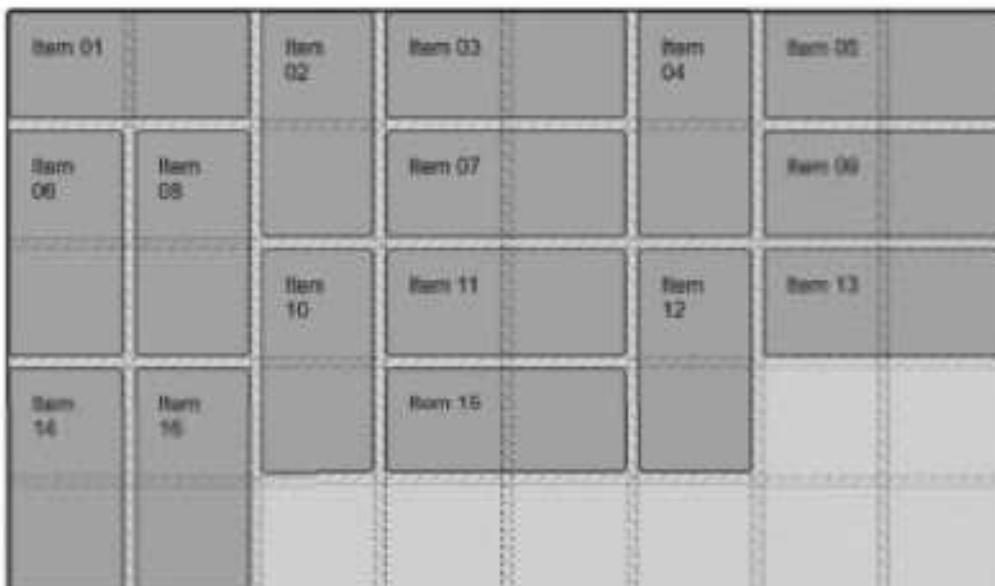
In gridlay-out en flexbox kan de plaatsingsvolgorde van items worden veranderd met de eigenschap order. Deze eigenschap krijgt een getal dat de rangorde bij de plaatsing aangeeft. Hoe lager het getal, hoe eerder het item wordt geplaatst. Een groot risico van plaatsen in een andere volgorde dan de HTML-flow is dat de tabvolgorde niet meer klopt. Voor gebruikers die afhankelijk zijn van toetsenbordbediening kan dat een groot probleem zijn.

De gaten in het raster opvullen

De eigenschap grid-auto-flow heeft nog een derde waarde: dense. Deze waarde kan los worden gebruikt of in combinatie met row of column. Met dense wordt de rasteritems opnieuw gerangschikt om zo veel mogelijk gaten in het raster op te vullen. In de afbeeldingen is het resultaat van de code te zien.



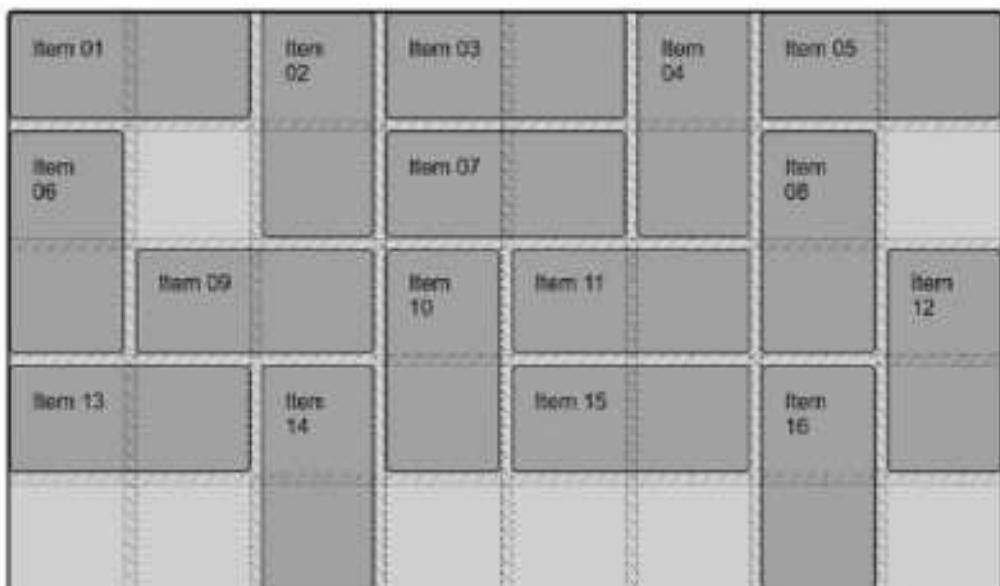
Afbeelding 10.48 De normale plaatsing van de items (10_48.html).



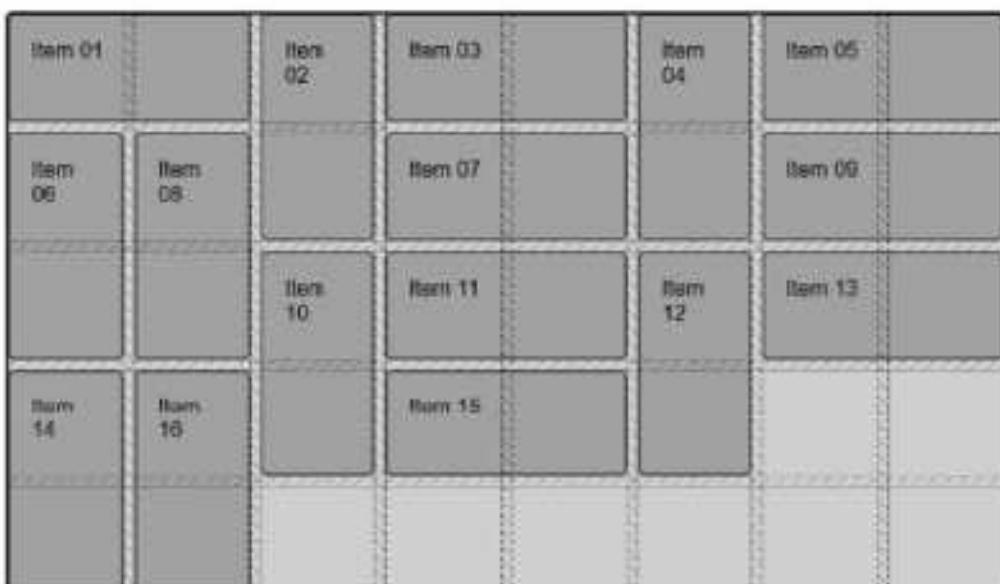
Afbeelding 10.49 De items zijn geplaatst met grid-auto-flow: dense (10_48.html).

De HTML is niet meer dan een rastercontainer met zestien items. Het enige verschil tussen de afbeeldingen is de vetgedrukte CSS-regel.

```
.grid {
    display: grid;
    grid-gap: 10px;
    grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));
    grid-auto-rows: 100px;
    grid-auto-flow: dense;
```



Afbeelding 10.48 De normale plaatsing van de items (10_48.html).



Afbeelding 10.49 De items zijn geplaatst met grid-auto-flow: dense (10_48.html).

De HTML is niet meer dan een rastercontainer met zestien items. Het enige verschil tussen de afbeeldingen is de vetgedrukte CSS-regel.

```
.grid {
    display: grid;
    grid-gap: 10px;
    grid-template-columns: repeat(auto-fill, minmax(100px, 1fr));
    grid-auto-rows: 100px;
    grid-auto-flow: dense;
```

```

}
.grid > *:nth-child(odd) {
  grid-column: span 2;
}
.grid > *:nth-child(even) {
  grid-row: span 2;
}

```

Uitlijnen in gridlay-out

In gridlay-out zijn er eigenschappen voor uitlijning van de tracks in de container, de items in het raster en afzonderlijke items in het raster. Elk van die onderdelen kan worden uitgelijnd in de rijrichting (`justify-*`) en in de blokrichting (`align-*`). (Zie ook de paragraaf *De toekomst: box alignment*.)



Het gaat om de items, niet om de inhoud van de items

Ook de uitlijning heeft alleen betrekking op de rasteritems, dus de kinderen van een rastercontainer. De tekst die in de afbeeldingen is te zien hoort daar dus niet bij. De inhoud van de items wordt uitgelijnd door eigenschappen ingesteld op die items. In de voorbeelden is flexbox gebruikt om de tekst te centreren in de items.

Alle items worden uitgelijnd in de rijrichting met `justify-items`. Dit wordt ingesteld op de rastercontainer. Afzonderlijke items worden uitgelijnd met `justify-self` en dit wordt uiteraard ingesteld op het rasteritem. De waarden zijn:

- auto
- normal
- start
- end
- center
- stretch (standaardwaarde)
- baseline
- first baseline
- last baseline

Afbeelding 10.50 toont voorbeelden van de meestgebruikte waarden. `auto`, `normal` en `stretch` geven hetzelfde resultaat.

Alle items worden uitgelijnd in de blokrichting met `align-items`. Dit wordt ingesteld op de rastercontainer. Afzonderlijke items worden uitgelijnd met `align-self` en dit wordt uiteraard ingesteld op het rasteritem. De waarden zijn hetzelfde als bij `justify-*`:



Afbeelding 10.50 Uitlijning in de rijrichting (`justify-*`) (10_50.html).



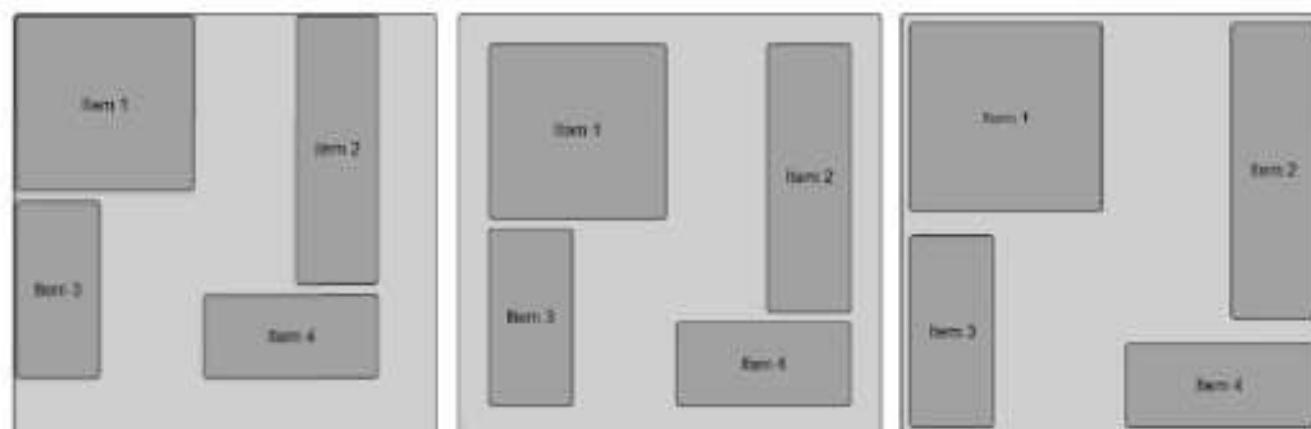
Afbeelding 10.51 Uitlijning in de blokrichting (`align-*`) (10_51.html).

- auto
- normal
- start
- end
- center
- stretch (standaardwaarde)
- baseline
- first baseline
- last baseline

Als de tracks minder ruimte gebruiken dan er in de container is, kunnen de tracks in die container worden uitgelijnd met `justify-content` en `align-content`. Dit wordt ingesteld op de container. Hiervoor zijn de volgende waarden beschikbaar:

- `normal`
- `start` (standaardwaarde)
- `end`
- `center`
- `stretch`
- `space-around`
- `space-between`
- `space-evenly`
- `baseline`
- `first baseline`
- `last baseline`

Standaard staat het raster altijd linksboven in de hoek van de container; de standaardwaarden van `justify-content` en `align-content` zijn dan ook `start`. Door beide in te stellen op `center` worden alle tracks in het midden van de container geplaatst; zo wordt het hele raster gecentreerd. Dat het echt om het uitlijnen van de tracks gaat en niet om het raster als geheel, is te zien met de waarde `space-around`.



Afbeelding 10.52 Van links naar rechts: normale plaatsing, alle tracks gecentreerd in de container en vrije ruimte verdeeld random de tracks (10_52.html).

Toepassing: nos.nl

De website nos.nl is een mooie toepassing voor gridlay-out. De site is gebouwd met flexbox, maar dat komt meer doordat flexbox eerder beschikbaar was dan dat het geschikter is dan gridlay-out. Flexbox werkt prima, dat zien de vele bezoekers dagelijks, maar de opbouw is typisch een klusje voor grid. In deze paragraaf wordt de homepage globaal nagebouwd met gridlay-out. Het gaat

om de hoofdlijnen, zodat u kunt zien hoe gridlay-out kan worden toegepast. Het is ook geen definitieve oplossing, maar gewoon een mogelijkheid. Hier wordt alleen de indeling van de pagina besproken, de overige opmaak staat in een afzonderlijke stylesheet.

Het voorbeeld is ook geschikt gemaakt voor mobiel en daarom komt u in de code media queries tegen. Kijk voor uitleg in de paragraaf *Responsive design*.



Afbeelding 10.53 De nieuwssite nos.nl op maximale breedte.

De vereenvoudigde remake bestaat globaal uit drie hoofdblokken:

- de header bevat het logo, het hoofdmenu en servicerubrieken zoals weer, verkeer en zoeken, en staat vast aan de bovenkant van het scherm;
- een of twee schermbrede foto's;
- de hoofdinhoud met in verschillende subblokken de nieuwsberichten;
- de footer met opnieuw het menu, informatie over de site en links naar sociale media.

Deze hoofdblokken zijn altijd in deze volgorde gestapeld en dat gaat vanzelf in de normale flow. Daar is geen lay-outmethode voor nodig. De meeste inhoud is beperkt tot een breedte van 1200px en wordt gecentreerd op de pagina, maar de twee foto's van het belangrijkste nieuws zijn samen altijd vensterbreed. Ook de header en de footer hebben een achtergrondkleur over de volle breedte, maar de inhoud is maximaal 1200px breed en gecentreerd. Dat gegeven maakt de lay-out net iets lastiger, want daardoor is het niet mogelijk om eenvoudig een groot inhoudsblok te maken met een breedte van 1200px en automatische marges waardoor het altijd in het midden van het venster staat.

Eerst de hoofdlijnen van de HTML. De complete code is te vinden in de bestanden 10_54.html en 10_54.css.

```
<body>
<header>
  <div class="logo"><span>n</span><span>o</span><span>s</span><span>.n</span></div>
  <h1>header van nos.nl</h1>
</header>
<section class="breaking">
  <div class="breaking-item">
    <h2>Breaking item 1 met een kop over twee regels</h2>
  </div>
  <div class="breaking-item">
    <h2>Breaking item 2 met een kop die ook wat langer is</h2>
  </div>
</section>
<main>
  <section class="nieuws">
    <ul class="nieuws-items">
      <li class="nieuws-item">
        
        <h2>Item kop</h2>
        <p>De tekst van een item is meestal niet zo lang. De rest van het bericht zit achter de link.</p>
        ... de rest van de nieuwsitems ...
      </li>
    </ul>
    <ul class="laatste-items">
      <li class="laatste-item">
        <p>Dit is nieuws dat het laatst is toegevoegd.</p>
        
      </li>
      ... de rest van de items ...
    </ul>
  </section>
</main>
<footer>
  <p>footer van nos.nl</p>
</footer>
</body>
```

Hierna volgt de complete CSS voor de lay-out. Deze wordt in de volgende paragrafen besproken. Overige opmaak wordt hier buiten beschouwing gelaten, maar is wel gekoppeld aan het bestand 10_54.html..

Eerst de hoofdlijnen van de HTML. De complete code is te vinden in de bestanden 10_54.html en 10_54.css.

```
<body>
<header>
  <div class="logo"><span>N</span><span>O</span><span>S</span></div>
  <h1>header van nos.nl</h1>
</header>
<section class="breaking">
  <div class="breaking-item">
    <h2>Breaking item 1 met een kop over twee regels</h2>
  </div>
  <div class="breaking-item">
    <h2>Breaking item 2 met een kop die ook wat langer is</h2>
  </div>
</section>
<main>
  <section class="nieuws">
    <ul class="nieuws-items">
      <li class="nieuws-item">
        
        <h2>Item kop</h2>
        <p>De tekst van een item is meestal niet zo lang. De rest van het bericht zit achter de link.</p>
        ... de rest van de nieuwsitems ...
      </li>
    </ul>
    <ul class="laatste-items">
      <li class="laatste-item">
        <p>Dit is nieuws dat het laatst is toegevoegd.</p>
        
      </li>
      ... de rest van de items ...
    </ul>
  </section>
</main>
<footer>
  <p>footer van nos.nl</p>
</footer>
</body>
```

Hierna volgt de complete CSS voor de lay-out. Deze wordt in de volgende paragrafen besproken. Overige opmaak wordt hier buiten beschouwing gelaten, maar is wel gekoppeld aan het bestand 10_54.html..

```

}
.news-items {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(176px, 1fr));
  gap: 1rem;
}
.ultimo-item {
  display: none;
}
@media screen and (min-width: 760px) {
  .ultimo-item {
    display: block;
  }
}
@media screen and (min-width: 840px) {
  .news-items {
    grid-template-columns: repeat(auto-fit, minmax(243px, 1fr));
  }
}
.ultimo-item {
  display: grid;
  grid-template-columns: 1fr 60px;
  gap: 8px;
}

```

Header en footer

De boxen van de header en de footer moeten de volle breedte vullen met een achtergrondkleur, terwijl de inhoud maximaal 1200px breed en gecentreerd is.

```

header,
footer {
  width: 100%
  display: grid;
  grid-template-columns: 1fr minmax(auto, 1200px) 1fr;
  grid-template-areas: ". content .";
  align-items: center;
  justify-items: center;
}
.logo {
  grid-area: content;
  justify-self: start;
}
h1 {
  grid-area: content;
}

```

```

justify-self: end;
}

footer > p {
  grid-area: content;
}

```

We combineren de elementselectors `header` en `footer` en stellen allereerst de breedte in op 100%. Daardoor zijn de header en de footer altijd net zo breed als hun container, in dit geval `body`. Vervolgens wordt een raster ingesteld van drie flexibele kolommen. De middelste kolom moet breed genoeg zijn voor de inhoud, maar niet breder dan 1200px. De twee buitenste kolommen zijn voor de ‘marges’. Die bestaan elk uit een gelijk deel van de beschikbare ruimte (1fr). Beschikbaar wil zeggen: de vensterbreedte min de 1200px voor de inhoud. Wordt het venster smaller dan 1200px, dan is de breedte van de buitenste kolommen dus (kleiner dan) 0. Met deze rasterdeclaratie zijn de header en de footer responsive zonder dat er media queries nodig zijn.

Dan worden de gebieden gedefinieerd:

```

grid-template-areas: ". content ."
align-items: center;
justify-items: center;

```

Er is één (impliciete) rij met drie kolommen: twee lege (de punten) en in het midden het gebied `content`. De inhoud van de rij wordt gecentreerd in de blokrichting en in de rijrichting.

De rasteritems zijn de klasse `.logo`, de kop `h1` en de tekstalinea in de footer. Ze worden alle drie geplaatst in het gebied `content`. Om te voorkomen dat in de header `.logo` en `h1` elkaar overlappen, worden ze met `justify-self` uitgelijnd naar het begin en eind van de rij.



Bovenaan gefixeerd

De header is gepositioneerd met `position: fixed`, waardoor deze altijd bovenaan zichtbaar blijft. Andere inhoud schuift er onderdoor. Om te voorkomen dat de header de erop volgende inhoud bij het laden van de pagina afdekt, krijgt de eerstvolgende sectie een bovenmarge gelijk aan de hoogte van de header (plus eventuele witruimte). Dat is hier de sectie met de klasse `.breaking`.

De sectie breaking

De sectie `.breaking` heeft als kenmerk dat de foto's op mobiel de breedte vullen en onder elkaar staan, en op een breder scherm naast elkaar staan en

dan samen de breedte vullen. Ze zijn dus breder dan de nieuwssectie. Met gridlayout is dit geen probleem:

```
.breaking {  
    display: grid;  
    grid-template-columns: repeat(auto-fit, minmax(360px, 1fr));  
}
```

Er worden zo veel mogelijk kolommen gemaakt met een breedte van minimaal 360px en maximaal de beschikbare ruimte (360px is de breedte van de kleinste gangbare mobiele telefoon). Op een breed scherm worden er bij deze instelling veel te veel kolommen gemaakt, terwijl er maar inhoud is voor twee kolommen. Dankzij auto-fit worden lege kolommen ingevouwen en wordt de vrijgekomen ruimte verdeeld over de twee foto's. Daardoor is ook deze sectie responsive zonder media queries!



Flexbox voor de koppen

De koppen in .breaking hebben nog als opmaakkenmerk dat die wel binnen de 1200px moeten blijven. Ze zijn uitgelijnd met de header en de nieuwsberichten. Dit is opgelost met flexbox. Hoe dat werkt, wordt uitgelegd aan het eind van de paragraaf *Lay-out: flexbox*.

De sectie main

De sectie main (maar eigenlijk de hele pagina) wordt *mobile first* opgebouwd. In goed opgebouwde HTML vallen op een klein scherm de hoofdonderdelen van de pagina vaak al vanzelf op hun plek. Door het scherm geleidelijk breder te maken, is te zien wanneer de lay-out niet meer klopt of niet overeenkomt met het ontwerp. Vanaf dat punt wordt de lay-out aangepast. Dat kan met media queries, met een flexibel raster, met een combinatie van beide en met nog meer methoden voor responsive design (zie de paragraaf *Responsive design*).

In de vereenvoudigde NOS-pagina bestaat main uit twee onderdelen:

- een sectie nieuws met een lijst van berichten waarvan er afhankelijk van de ruimte twee of drie naast elkaar staan;
- een sectie met een lijst van kort laatste nieuws, die naast de nieuwssectie staat, maar alleen zichtbaar is op een scherm van minimaal 760px breed.

De sectie nieuws

Op een smal scherm bestaat de sectie nieuws uit één kolom. Daar is geen CSS voor nodig. (Verwar nieuws niet met nieuws-items, dat zijn wel twee kolommen. Nieuws is de (raster)container van nieuws-items.)

dan samen de breedte vullen. Ze zijn dus breder dan de nieuwssectie. Met gridlay-out is dit geen probleem:

```
.breaking {  
    display: grid;  
    grid-template-columns: repeat(auto-fit, minmax(360px, 1fr));  
}
```

Er worden zo veel mogelijk kolommen gemaakt met een breedte van minimaal 360px en maximaal de beschikbare ruimte (360px is de breedte van de kleinste gangbare mobiele telefoon). Op een breed scherm worden er bij deze instelling veel te veel kolommen gemaakt, terwijl er maar inhoud is voor twee kolommen. Dankzij auto-fit worden lege kolommen ingevouwen en wordt de vrijgekomen ruimte verdeeld over de twee foto's. Daardoor is ook deze sectie responsive zonder media queries!



Flexbox voor de koppen

De koppen in `.breaking` hebben nog als opmaakkenmerk dat die wel binnen de 1200px moeten blijven. Ze zijn uitgelijnd met de header en de nieuwsberichten. Dit is opgelost met flexbox. Hoe dat werkt, wordt uitgelegd aan het eind van de paragraaf *Lay-out: flexbox*.

De sectie main

De sectie main (maar eigenlijk de hele pagina) wordt *mobile first* opgebouwd. In goed opgebouwde HTML vallen op een klein scherm de hoofdonderdelen van de pagina vaak al vanzelf op hun plek. Door het scherm geleidelijk breder te maken, is te zien wanneer de lay-out niet meer klopt of niet overeenkomt met het ontwerp. Vanaf dat punt wordt de lay-out aangepast. Dat kan met media queries, met een flexibel raster, met een combinatie van beide en met nog meer methoden voor responsive design (zie de paragraaf *Responsive design*).

In de vereenvoudigde NOS-pagina bestaat main uit twee onderdelen:

- een sectie nieuws met een lijst van berichten waarvan er afhankelijk van de ruimte twee of drie naast elkaar staan;
- een sectie met een lijst van kort laatste nieuws, die naast de nieuwssectie staat, maar alleen zichtbaar is op een scherm van minimaal 760px breed.

De sectie nieuws

Op een smal scherm bestaat de sectie nieuws uit één kolom. Daar is geen CSS voor nodig. (Verwar nieuws niet met nieuws-items, dat zijn wel twee kolommen. Nieuws is de (raster)container van nieuws-items.)

Pas vanaf 760px moeten er twee kolommen zijn. Daarom staat de code in een media query:

```
media screen and (min-width: 760px) {
    .nieuws {
        display: grid;
        grid-template-columns: 1fr 300px;
        max-width: 1200px;
        margin: auto;
    }
}
```

Er wordt een raster ingesteld met een flexibele kolom van 1fr en een vaste kolom van 300px. De maximale totale breedte van 1200px is afzonderlijk ingesteld, maar dat zou hier ook in de rasterdefinitie kunnen:

```
grid-template-columns: minmax(0, 900px) 300px;
```

Hiermee wordt het blok ook niet breder dan 1200px ($900 + 300$). Het is een kwestie van voorkeur. Het blok wordt gecentreerd op de pagina met de automatische marge.

De lijsten nieuws-items en laatste-items

De lijst `nieuws-items` bestaat op een smal scherm uit één kolom. Pas als er genoeg ruimte is voor een veelvoud van 176px plus de tussenruimte van 1rem komen er meer flexibele kolommen.

```
.nieuws-items {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(176px, 1fr));
    gap: 1rem;
}
```

De lijst `laatste-items` wordt op een smal scherm niet weergegeven:

```
.laatste-items {
    display: none;
}
```

Vanaf 760px komt `laatste-items` erbij.

```
media screen and (min-width: 760px) {
    .laatste-items {
        display: block;
    }
}
```

Merk op dat er geen kolommen, rijen of gebieden worden ingesteld. nieuws-items en laatste-items worden geplaatst volgens de HTML-flow en komen daardoor vanzelf in de goede kolom terecht.

Om te voorkomen dat bij grotere breedten vier kolommen nieuws-items ontstaan, wordt nog een media query toegevoegd. De 840px is gekozen omdat er dan een naadloze overgang is naar grotere items.

```
@media screen and (min-width: 840px) {
    .nieuws-items {
        grid-template-columns: repeat(auto-fit, minmax(243px, 1fr));
    }
}
```

De componenten nieuws-item en laatste-item

Elk afzonderlijk nieuws-item bestaat eenvoudig uit gestapelde blokken. Daar komt geen lay-outstijl aan te pas (wel overige opmaak).

Elk laatste-item is opgebouwd met gridlay-out, want dat maakt het eenvoudig om de tekst en de afbeelding naast elkaar te zetten:

```
.laatste-item {
    display: grid;
    grid-template-columns: 1fr 60px;
    gap: 8px;
}
```

Twee kolommen, een vaste afmeting voor de afbeelding en een flexibele kolom voor de tekst, en 8px tussenruimte. Meer is het niet.



Afbeelding 10.54 Nieuws-item en laatste-item (10_54.html).

Lay-out: flexbox

Flexbox was net wat eerder productiever dan gridlay-out en bepaalt daardoor nu op veel weppagina's de indeling. Daar is het niet per se de beste oplossing voor. De kracht van flexbox zit meer in de vormgeving van componenten. Nog even het belangrijke verschil met gridlay-out: flexbox draait om het *verdelen van inhoud in één richting*, terwijl grid *inhoud op een raster plaatst en in twee richtingen tegelijk* werkt. De overeenkomsten zijn de flexibiliteit, de uitlijning in alle richtingen en de mogelijkheid om de weergavevolgorde te laten afwijken van de volgorde van de HTML-bron. Flexbox is vooral handig voor componenten die mogen groeien en krimpen in de beschikbare ruimte. Flexbox doet precies dat: flexibele boxen maken.

Kenmerken van flexbox

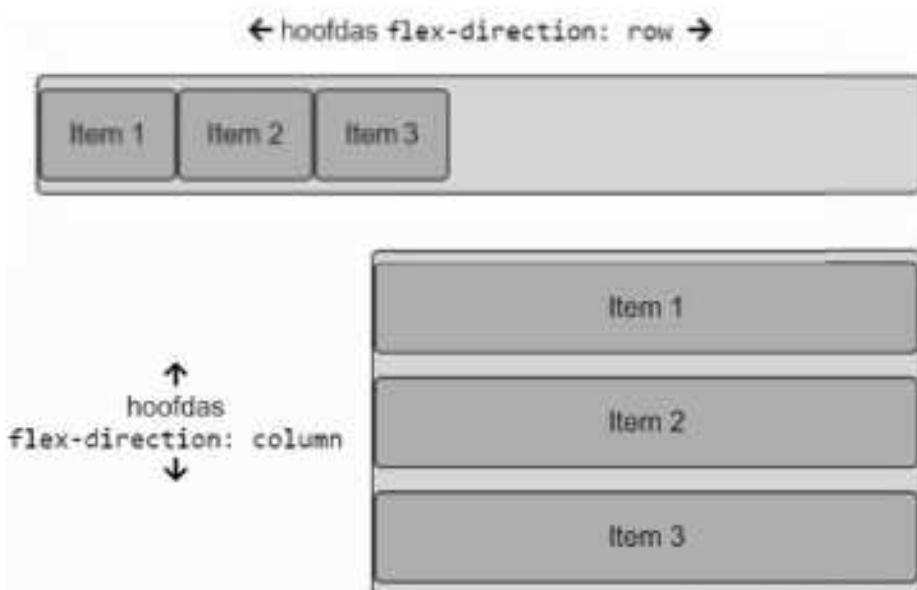
Enkele kenmerken van flexbox zijn:

- items kunnen in één richting worden geplaatst: óf de rijrichting (van links naar rechts) óf de blokrichting (van boven naar beneden);
- de plaatsingsrichting kan worden omgedraaid;
- de volgorde van items kan worden veranderd;
- items kunnen op een regel/kolom naast elkaar blijven staan of omlopen naar de volgende regel/kolom;
- items kunnen groeien en krimpen naar gelang de ruimte;
- items kunnen in verschillende verhoudingen groeien en krimpen;
- items kunnen worden uitgelijnd ten opzichte van hun container of elkaar.

Het assenstelsel van flexbox

Het assenstelsel van flexbox is gebaseerd op de begrippen hoofdas en kruisas (afbeelding 10.55). De hoofdas is de as waarlangs items worden geplaatst en het is de enige richting waarin flexbox flexibel is. Vandaar ook dat het een een-dimensionaal systeem wordt genoemd. Gridlay-out is tweedimensionaal, want daarmee worden items tegelijk in rijen en kolommen geplaatst.

De kruisas staat haaks op de hoofdas en is de as waarlangs items kunnen omlopen als de regel op de hoofdas vol is. De hoofdas loopt standaard in de schrijfrichting. In het Nederlands (en Engels en alle andere op het Latijn gebaseerde systemen) loopt de hoofdas dus van links naar rechts. Dan loopt de kruisas van boven naar beneden. De richting van de hoofdas kan worden aangepast.



Afbeelding 10.55 De hoofdas is de enige as waarlangs items flexibel zijn (10_55.html).



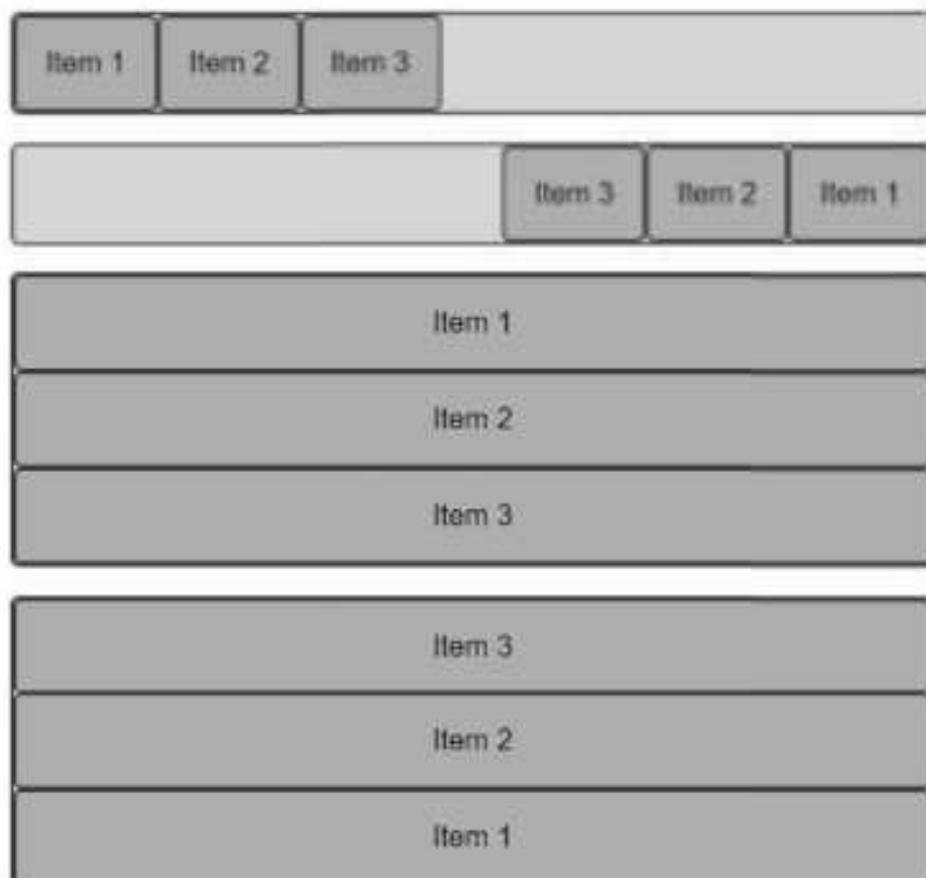
Groeien in de kruisrichting

In de afbeelding zijn bij `flex-direction: column` de items breder, maar dat komt niet door de groei-eigenschap van flexitems. De hoofdas is de flexibele richting en die loopt hier van boven naar beneden. Het groeien in de breedte (de kruisas) komt doordat flexitems in de kruisrichting standaard worden opgerekt (`stretch`) tot containerbreedte. Dit wordt uitgelegd in de paragraaf *Uitlijnen in de kruisrichting*. Dat de items bij `column` niet hoger en bij `flex-direction: row` niet breder worden, is juist wel een groeikenmerk van flexbox. De standaardinstelling is namelijk niet groeien.

Flex-direction

De hoofdas wordt bepaald door de eigenschap `flex-direction` (afbeelding 10.56). Deze eigenschap kan een van de volgende vier waarden hebben:

- `row` (standaard) plaatst items in de rijrichting;
- `row-reverse` plaatst omgekeerd in de rijrichting;
- `column` plaatst de flexitems in de blokrichting;
- `column-reverse` plaatst omgekeerd in de blokrichting.



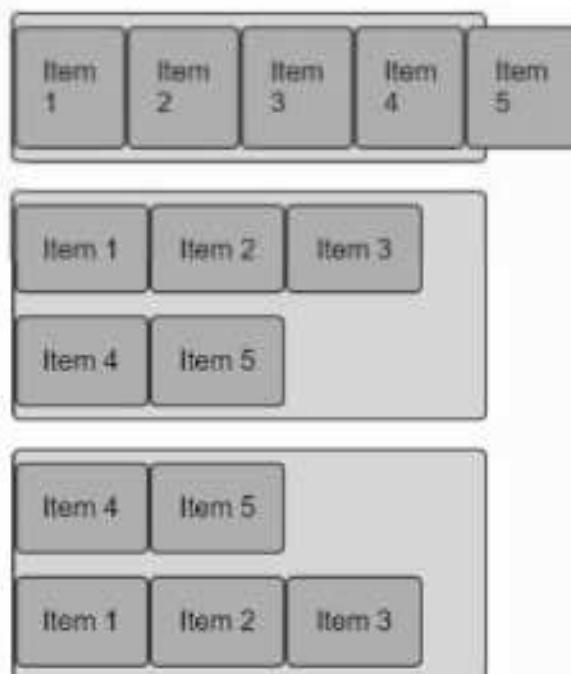
Afbeelding 10.56 Flex-direction met achtereenvolgens de waarden `row`, `row-reverse`, `column` en `column-reverse` (let op de nummering) (10_56.html).

Flex-wrap

Standaard worden alle items in de hoofdrichting achter elkaar geplaatst, ook als daarbij overloop (en een schuifbalk) ontstaat. Doorgaan op de volgende 'regel' wordt ingesteld met de eigenschap `flex-wrap`.

Voor `flex-wrap` zijn drie waarden beschikbaar:

- `nowrap` (standaard) houdt alle items op één regel in de hoofdrichting;
- `wrap` laat items doorlopen op de volgende regel in de kruisrichting;
- `wrap-reverse` laat ook items doorlopen op de volgende regel in de kruisrichting, maar draait het begin en eind van de kruisrichting om. Daardoor staat de volgende regel voor de eerste.



Afbeelding 10.57 Flex-wrap met de instellingen nowrap, wrap en wrap-reverse (10_57.html).

Flex-flow

Met de eigenschap `flex-flow` worden `flex-direction` en `flex-wrap` ineens ingesteld. Een kolom met omloop wordt op de klasse `.flex` ingesteld als:

```
.flex {  
  flex-flow: column wrap;  
}
```

Het is niet verplicht om twee waarden te geven. De weggelaten waarde krijgt de standaardinstelling `row` of `no-wrap`.

Flexcontainers en flexitems

Om flexbox te kunnen gebruiken moet eerst een flexibele omgeving worden gemaakt: de *flex formatting context*. Daarvoor heeft de eigenschap `display` twee waarden:

- `flex` een element genereert een flexcontainer op blokniveau;
- `inline-flex` een element genereert een flexcontainer op regelniveau.

De flexcontainer krijgt de eigenschap `display: flex`. Alleen de kindelementen van een flexcontainer zijn *flexitems*. De eigenschappen `flex-direction` en `flex-wrap` (of `flex-flow`) bepalen hoe de flexitems worden geplaatst.

Zodra er een flexcontainer is gemaakt, hebben de kindelementen de standaardeigenschappen van flexitems:

- het item mag niet groeien;
- het item mag wel krimpen;
- de grootte in de hoofdrichting is afhankelijk van de inhoud of de ingestelde breedte/hoogte.

Items flexibel maken met de eigenschap flex

Flexitems kunnen groeien of krimpen om de beschikbare ruimte op te vullen. Per item kan worden ingesteld hoeveel het mag groeien of krimpen in relatie tot de andere items. Deze flexibele lengte wordt bepaald door de eigenschap `flex`. Zodra een element een flexitem is geworden, zijn de `width` of `height` nog maar een richtlijn voor de afmeting (afhankelijk van de hoofdrichting in `flex-direction: row` of `column`). De waarden van `flex` bepalen de uiteindelijke afmeting.



Beschikbare ruimte

Net als bij gridlay-out (de eenheid `fr`) is bij flexbox beschikbare ruimte een kernbegrip. Van de totale ruimte (de lengte op de hoofdas van de flexcontainer of een venster) worden eerst alle vast ingestelde lengtes van items afgetrokken. Wat overblijft is de beschikbare ruimte. Deze ruimte wordt verdeeld tussen de flexitems.

Hoewel afzonderlijke eigenschappen kunnen worden ingesteld, wordt aanbevolen om de verkorte notatie `flex` te gebruiken. Daarmee krijgen de niet-ingestelde eigenschappen automatisch optimale waarden. In `flex` zijn de eigenschappen `flex-grow`, `flex-shrink` en `flex-basis` samengevoegd.

De eigenschap `flex` kan de waarde `none` of `auto` hebben, of een of meer waarden voor de volgende eigenschappen:

- `flex-grow` met een nummer wordt de groefactor ingesteld. Beschikbare ruimte wordt verdeeld om het item te laten groeien relatief aan de andere items. Wordt deze waarde weggelaten, dan is die 1.
- `flex-shrink` met een nummer wordt de krimpfactor ingesteld, ook relatief aan de andere items. Hierbij gaat het om de verdeling van 'negatieve vrije ruimte', ruimtegebrek dus. Wordt deze waarde weggelaten, dan wordt die 1. De krimpfactor wordt vermenigvuldigd met `flex-basis`.
- `flex-basis` is de basisafmeting van de flexitems waarop de groei en krimp worden gebaseerd. Deze kan worden ingesteld op een lengte, percentage, `auto` of `content` (in feite gelijk aan `max-content`, dus de inhoud)

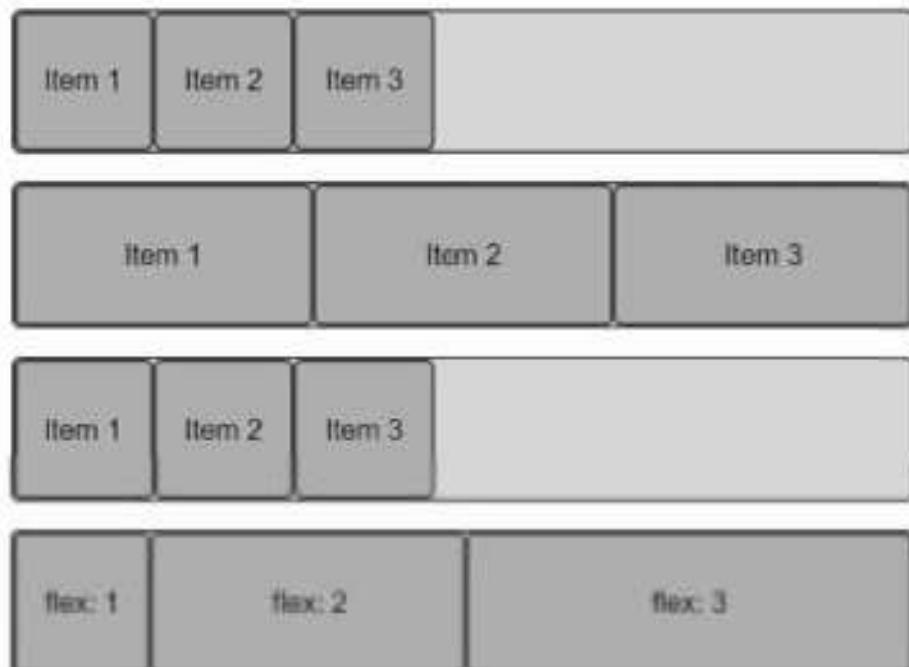
van het flexitem). auto haalt de waarde uit width of height, afhankelijk van de hoofdas. Wordt flex-basis weggelaten, dan is de waarde 0.

Instellingen voor flex

Dit zijn vooraf gemaakte instellingen die in veel praktijksituaties voldoen:

- **flex: initial** dit is de standaardinstelling: `flex: 0 1 auto`. Het item groeit niet, mag wel krimpen en de basisafmeting wordt bepaald door de lengte op de hoofdas (width of height, die kan auto zijn).
- **flex: auto** hiermee wordt beschikbare ruimte in de flexcontainer volledig benut. Groeien en krimpen gebeuren naar behoefte. Dit is hetzelfde als `flex: 1 1 auto`.
- **flex: none** met deze instelling krijgt het item op de hoofdas de waarde van width/height (kan auto zijn): er is geen groei en geen krimp. Het item is dus niet flexibel.
- **flex: positief getal** het item groeit met de vrije ruimte in de container. Dit is hetzelfde als `flex: getal 1 0`. Het getal geeft het proportionele deel van de vrije ruimte aan. Als alle items hetzelfde getal hebben, krijgen ze een gelijk deel van de vrije ruimte. Door aan flexitems verschillende waarden toe te wijzen, krijgen ze een groter of kleiner deel van de vrije ruimte.

Flexitems worden nooit smaller dan het kleinste deel van de inhoud (een woord), maar ze kunnen een vaste minimale breedte (en hoogte) krijgen met een lengtewaarde voor min-width en min-height.



Afbeelding 10.58 Het resultaat van de flex-waarden initial, auto, none en flex: 1, 2 en 3 ([h10_58.html](#)).

Samenvatting: het formaat van flexitems

Het is belangrijk om te beseffen dat het formaat van flexitems afhangt van drie dingen:

- de hoofdas, want in de kruisas krijgt het item altijd de afmeting volgens het boxmodel;
- de factor waarmee het item mag groeien of krimpen in relatie tot de groei-/krimpfactor van andere items;
- de basisafmeting van het item.

In de volgende afbeelding is de hoofdrichting `row`. De breedte van de items is `auto` (er is geen `width` ingesteld) en in de eerste twee regels hebben ze verschillende groelfactoren: 1, 4, 2 en 100, 400, 200. De basisafmeting van de items wordt niet ingesteld en is daarom 0. In de onderste regel zijn alle groelfactoren 1, maar is de basisafmeting 100, 400, 200.

```
<div class="flex">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>
<div class="flex plus">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>
<div class="flex basis">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>
/* CSS */
.flex {
  display: flex;
}
.flex > *:nth-child(1) {
  flex: 1;
}
.flex > *:nth-child(2) {
  flex: 4;
}
.flex > *:nth-child(3) {
  flex: 2;
}
```

Samenvatting: het formaat van flexitems

Het is belangrijk om te beseffen dat het formaat van flexitems afhangt van drie dingen:

- de hoofdas, want in de kruisas krijgt het item altijd de afmeting volgens het boxmodel;
- de factor waarmee het item mag groeien of krimpen in relatie tot de groei-/krimpfactor van andere items;
- de basisafmeting van het item.

In de volgende afbeelding is de hoofdrichting row. De breedte van de items is auto (er is geen width ingesteld) en in de eerste twee regels hebben ze verschillende groefactoren: 1, 4, 2 en 100, 400, 200. De basisafmeting van de items wordt niet ingesteld en is daarom 0. In de onderste regel zijn alle groefactoren 1, maar is de basisafmeting 100, 400, 200.

```
<div class="flex">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>

<div class="flex plus">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>

<div class="flex basis">
  <div>Item 1</div>
  <div>Item 2</div>
  <div>Item 3</div>
</div>

/* CSS */
.flex {
  display: flex;
}
.flex > *:nth-child(1) {
  flex: 1;
}
.flex > *:nth-child(2) {
  flex: 4;
}
.flex > *:nth-child(3) {
  flex: 2;
}
```

```

.plus > *:nth-child(1) {
  flex: 100;
}
.plus > *:nth-child(2) {
  flex: 400;
}
.plus > *:nth-child(3) {
  flex: 200;
}
.basis > *:nth-child(1){
  flex: 1 1 100px;
}
.basis > *:nth-child(2){
  flex: 1 1 400px;
}
.basis > *:nth-child(3){
  flex: 1 1 200px;
}

```



Afbeelding 10.59 Hoe hoog de groeifactor is doet er niet toe, het gaat om de verhouding met de groeifactoren van andere items. In de onderste rij bepaalt de basisafmeting van 200px de groeiverhouding (10_59.html).

Items uitlijnen met automarge

Items in een flexcontainer kunnen op de hoofdas en de kruis-as op twee manieren worden uitgelijnd:

- automatische marges zoals in het boxmodel;
- sleutelwoorden voor uitlijning op de assen.

Flex-items uitlijnen met behulp van automatische marges komt overeen met het uitlijnen in het boxmodel. Geef het flex-item eenvoudig de declaratie margin: auto en het wordt horizontaal en verticaal gecentreerd. Met automatische marge op een van de richtingen wordt het item weggeduwd naar de andere richting.

Let op: zodra flexitems mogen groeien, blijft er geen ruimte over voor marges. Voor de afbeeldingen is flex niet ingesteld en dus geldt de standaardwaarde flex: 0 1 auto. Dat betekent: niet groeien, wel krimpen en de basisgrootte is de width van het element of auto.



Afbeelding 10.60 Een component centreren in zijn container was nog nooit zo eenvoudig (10_60.html).



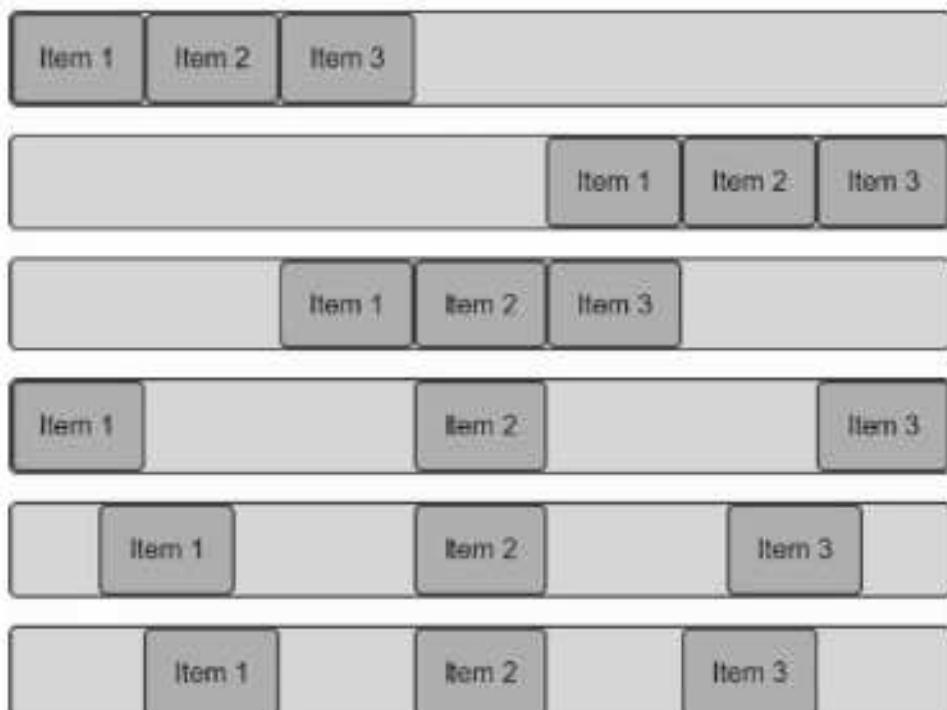
Afbeelding 10.61 Boven twee items met rondom automatische marges. Bij de onderste items zijn de marges gereset naar 0, behalve rechts en links (10_60.html).

Uitlijnen op de hoofdas: justify-content

De eigenschap `justify-content` wordt ingesteld op de flexcontainer en zorgt voor de uitlijning van de items op de hoofdas. De buitenkant van de marge is de buitenste rand van het flexitem. De volgende waarden zijn mogelijk:

- `flex-start` alle items worden uitgelijnd naar het begin van de regel (standaard);
- `flex-end` alle items worden uitgelijnd naar het eind van de regel;
- `center` items worden vanuit het midden tegen elkaar aan gezet;

- `space-between` vrije ruimte wordt tussen de items verdeeld. Het eerste en laatste item staat tegen de rand. Als er geen vrije ruimte of maar één item is, is het resultaat hetzelfde als `flex-start`;
- `space-around` vrije ruimte wordt tussen de items verdeeld en aan het begin en eind van de regel komt een halve vrije ruimte. Als er geen vrije ruimte of maar één item is, is het resultaat hetzelfde als `center`;
- `space-evenly` als `space-around`, maar vrije ruimte wordt evenredig verdeeld, ook aan de buitenkant.



Afbeelding 10.62 Uitlijnen met `justify-content: flex-start, flex-end, center, space-between, space-around en space-evenly` (10_62.html).

Als voorbeeld nemen we een navigatiebalk met links de koppelingen en rechts een zoekvak.

```
<nav>
  <ul>
    <li><a href="#">Links </a></li>
    <li><a href="#">Midden </a></li>
    <li><a href="#">Rechts</a></li>
    <li><label>Zoeken <input></label></li>
  </ul>
</nav>
```



Afbeelding 10.63 Alles op zijn plek. Links het menu, rechts het zoekvak (*10_63.html*).

In de CSS zijn maar twee dingen belangrijk: `` moet een flexcontainer zijn en het zoekvak moet een automatische marge links hebben. Die duwt het zoekvak naar de rechterkant en cuwt tegelijk de menu-items tegen de linkerkant aan.

```
ul {
    display: flex;
}
li:last-child {
    margin: 0 0 0 auto;
}
```

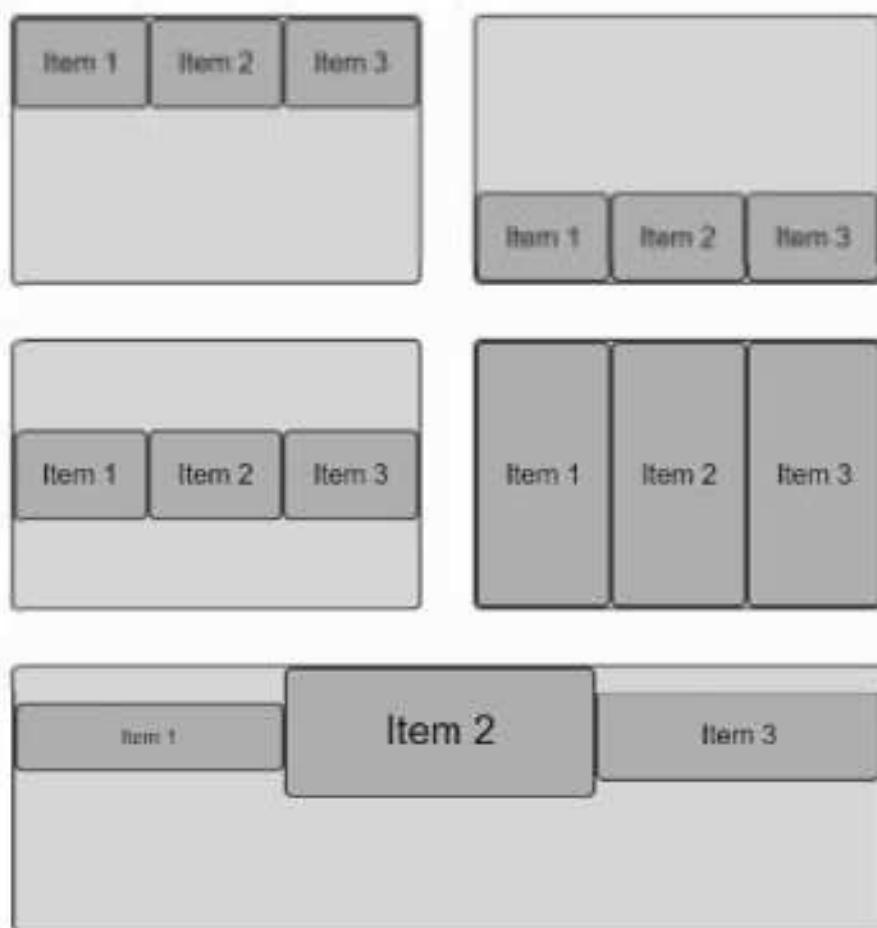
Uitlijnen op de kruisas: align-self

Voor uitlijning op de kruisas zijn er twee eigenschappen:

- `align-items` wordt ingesteld op de flexcontainer en bepaalt de uitlijning van alle flexitems in de container.
- `align-self` wordt ingesteld op afzonderlijke flexitems om die een andere uitlijning te geven dan de algemene.

De buitenkant van de marge is de buitenste rand van het flexitem. De waarden zijn voor beide eigenschappen bijna hetzelfde:

- `flex-start` het item wordt uitgelijnd op het begin van de kruisas;
- `flex-end` het item wordt uitgelijnd op het eind van de kruisas;
- `center` het item wordt gecentreerd op de kruisas;
- `baseline` alle items worden zo geplaatst dat hun basislijn gelijkloopt, waarbij het item met de grootste ruimte tussen basislijn en margerand tegen de starrand van de kruisas wordt geplaatst;
- `stretch` het flexitem wordt uitgerekt zodat het de hele ruimte op de kruisas vult (standaard);
- `auto` geldt alleen voor `align-self` en daarvoor is het de standaard-waarde. De werkelijke waarde wordt afgeleid van `align-items` van de flexcontainer.



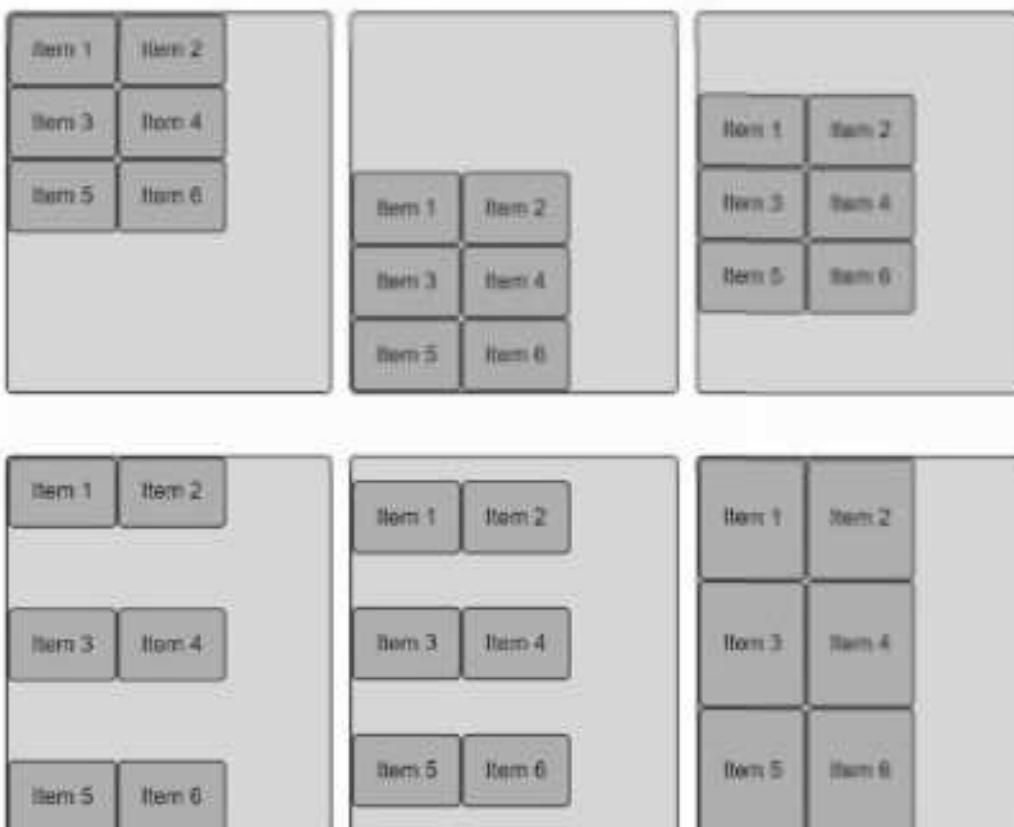
Afbeelding 10.64 Uitlijnen op de kruisas: `flex-start`, `flex-end`, `center`, `stretch` en `baseline`. Bij `baseline` zijn verschillende tekstgrootten gebruikt, zodat goed is te zien dat echt op de baseline wordt uitgelijnd (10_64.html).

Regels uitlijnen: align-content

In flexcontainers met meerdere regels (dan heeft de eigenschap `flex-wrap` de waarde `wrap`) kunnen de regels in de vrije ruimte op de kruisas worden uitgelijnd. Hiervoor wordt de eigenschap `align-content` gebruikt. Let op: als er maar één regel is, heeft deze eigenschap geen effect. De waarden zijn:

- `flex-start` alle regel worden samengepakt richting het begin van de flexcontainer;
- `flex-end` alle regel worden samengepakt richting het eind van de flexcontainer;
- `center` alle regel worden samengepakt rond het midden van de flexcontainer. De vrije ruimte aan het begin en eind is gelijk;
- `space-between` de regels worden verdeeld over de container. De eerste en de laatste staan tegen de rand en de vrije ruimte wordt verdeeld tussen de regels;

- `space-around`: de regels worden verdeeld over de container en de vrije ruimte wordt verdeeld tussen de regels. Voor de eerste en na de laatste regel komt een halve vrije ruimte;
- `stretch`: regels worden uitgerekt om de vrije ruimte op te vullen.



Afbeelding 10.65 Regels uitlijnen op de kruis-as. Van linksboven naar rechtsonder: `flex-start`, `flex-end`, `center`, `space-between`, `space-around` en `stretch` (10_65.html).

Toepassing: navigatiebalk

Hoewel flexbox kan worden gebruikt als rastersysteem voor complete pagina-lay-outs, is dat niet de beste toepassing. Gebruik daar vooral gridlay-out voor. Het menu voorbeeld in de paragraaf *Uitlijnen met automarges* is wel een typisch voorbeeld voor flexbox, net als het horizontaal en verticaal centreren van een item in dezelfde paragraaf. Een item centreren kan natuurlijk ook met `justify-content: center` en `align-items: center`, maar `margin: auto` is toch een stuk korter.

Variaties op de eerder gemaakte navigatiebalk zijn natuurlijk ook mogelijk. In het volgende voorbeeld zijn er drie onderdelen: links een menu, in het midden een logo dat een link is en rechts een zoekvak. Hier is nog verrassend veel over te vertellen.

```

<nav>
  <ul class="menu">
    <li><a href="">Menu 1</a></li>
    <li><a href="">Menu 2</a></li>
    <li><a href="">Menu 3</a></li>
  </ul>
  <a href=""><i class="fab fa-github"></i></a>
  <div class="zoekvak">
    <input type="text" placeholder="Wat zoek je?">
    <button type="submit"><i class="fas fa-search"></i></button>
  </div>
</nav>
/* CSS */
.flex {
  display: flex;
  justify-content: space-between;
}
.menu {
  display: flex;
}
.a {
  align-self: center;
}
.zoekvak {
  display: flex;
}

```



Afbeelding 10.66 Navigatiebalk met zoekvak en logo in het midden (10_66.html).



Pictogrammenlettertype

In de HTML staat twee keer een element `<i>`, namelijk `<i class="fab fa-github"></i>` en `<i class="fas fa-search"></i>`. Hiermee worden pictogrammen weergegeven uit het *icon font* (pictogrammenlettertype) Font Awesome. Met zulke bibliotheken kunnen eenvoudig mooie en consistentie pictogrammen op een website worden gebruikt. Zie voor uitleg de website fontawesome.com.

Analyse van de code

Eerst wordt van `<nav>` een flexcontainer gemaakt. Op de hoofdas (rij) wordt ruimte verdeeld tussen de items met `justify-content`. Daardoor staat het

menu links, het logo in het midden en het zoekvak rechts. De items worden op de kruis-as (kolom) net zo hoog als de flexcontainer dankzij de standaardinstelling `align-items: stretch`.

```
.nav {
  display: flex;
  justify-content: space-between;
}
```

De lijst `` is een flexitem, maar wordt zelf ook een flexcontainer. Deze heeft verder geen flexinstellingen, maar de menu-items worden nu vanzelf naast elkaar geplaatst dankzij de standaardinstelling `flex-direction: row`.

```
.ul {
  display: flex;
}
```

Om het logo `<i>` dat in de link `<a>` staat op de kruis-as te kunnen centreren, moet `<a>` worden gecentreerd:

```
a {
  align-self: center;
}
```

`<a>` is standaard net zo groot als `<i>`, waardoor het logo vanzelf middenin `<a>` staat. Maar `<a>` is een flexitem van `<nav>` en daarin is de standaardinstelling `align-items: stretch` van kracht. Daardoor wordt `<a>` net zo hoog als `<nav>`, maar dan staat het logo niet meer in het midden, want dat staat vanzelf tegen de bovenkant van `<a>` (en dus van `<nav>`). Dit wordt opgelost met `align-self: center`. Daardoor groeit `<a>` niet en staat het met het logo in het midden van `<nav>`.

Het laatste flexitem is het `.zoekvak`. Ook dat wordt een flexcontainer, zodat de `<input>` en de `<button>` netjes worden geplaatst. Die worden wel vanzelf naast elkaar gezet (de elementen maken regelboxen, geen blokboxen), maar omdat zij nu ook flexitems zijn die de standaardinstelling `align-items: stretch` hebben, worden ze nu ook net zo hoog als de flexcontainer.

```
.zoekvak {
  display: flex;
}
```

Toepassing: kaarten

Een kaart is een veelgebruikte component, meestal met een afbeelding, een korte tekst en een link. Er zijn allerlei toepassingen mogelijk: producten, films, boeken, personen enzovoort.

Productkaart

De productkaart in de afbeelding heeft een kop, een afbeelding, tekst en een link die eruitziet als een knop. De kaart heeft ook twee problemen: de knoppen staan niet op dezelfde hoogte en ze staan niet in het midden.



Afbeelding 10.67 Productkaarten met een knop die niet op dezelfde hoogte en niet in het midden staat (10_67.html).

De HTML voor een kaart is:

```
<div class="card">
  <h2>Lorem ipsum</h2>
  
  <p>... tekst ...</p>
  <a href="#">Lorem!</a>
</div>
```

De CSS bestaat voor een groot deel uit opmaak. De enige relevante instellingen voor de lay-out zijn de breedte en hoogte van de kaart. In de normale flow komt alles vanzelf onder elkaar te staan. Het is aan te raden om de afbeelding een breedte van 100% te geven. Daarmee krijgt die de breedte van de container en zo past de afbeelding altijd. De hoogte schaalt proportioneel mee.

Toepassing: kaarten

Een kaart is een veelgebruikte component, meestal met een afbeelding, een korte tekst en een link. Er zijn allerlei toepassingen mogelijk: producten, films, boeken, personen enzovoort.

Productkaart

De productkaart in de afbeelding heeft een kop, een afbeelding, tekst en een link die eruitziet als een knop. De kaart heeft ook twee problemen: de knoppen staan niet op dezelfde hoogte en ze staan niet in het midden.



Afbeelding 10.67 Productkaarten met een knop die niet op dezelfde hoogte en niet in het midden staat (10_67.html).

De HTML voor een kaart is:

```
<div class="card">
  <h2>Lorem ipsum</h2>
  
  <p> ... tekst ... </p>
  <a href="#">Lorem!</a>
</div>
```

De CSS bestaat voor een groot deel uit opmaak. De enige relevante instellingen voor de lay-out zijn de breedte en hoogte van de kaart. In de normale flow komt alles vanzelf onder elkaar te staan. Het is aan te raden om de afbeelding een breedte van 100% te geven. Daarmee krijgt die de breedte van de container en zo past de afbeelding altijd. De hoogte schaalt proportioneel mee.

```
.card {
    width: 300px;
    height: 550px;
    border: 1px solid #171717;
    border-radius: 8px;
    margin-left: 1rem;
}

h2 {
    color: #171717;
    margin-top: 0;
    padding: 8px 8px 0;
}

img {
    width: 100%;
}

p {
    padding: 0 8px 8px;
}

a {
    background-color: hsl(100, 75%, 50%);
    color: #171717;
    margin-bottom: 8px;
    padding: 8px 12px;
    border-radius: 8px;
}
```

Afbeelding 10.68 toont het gewenste resultaat. Daar is verrassend weinig extra CSS voor nodig. De `.card` wordt een flexcontainer met `column` als hoofdas, de `<p>` krijgt `flex: 1` en mag daardoor groeien (dit is hetzelfde als `flex: 1 1 0`, terwijl de standaardinstelling `flex: 0 1 auto` is). Doordat `<p>` groeit, wordt `<a>` omlaag geduwd. Tot slot wordt `<a>` gecentreerd op de kruis-as (in dit voorbeeld loopt die horizontaal).

```
.card {
    display: flex;
    flex-flow: column;
}

p {
    flex: 1;
}

a {
    align-self: center;
}
```



Afbeelding 10.68 Het eindresultaat (10_67.html).

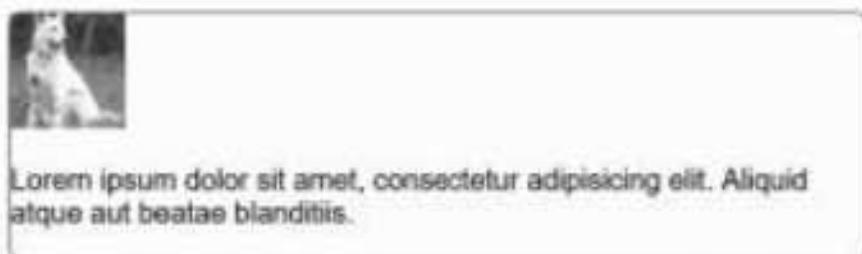
Afbeelding links/rechts

Een ander type kaart heeft een kleine afbeelding en een korte beschrijving. De HTML is:

```
<div class="card">
  
  <p> ...tekst... </p>
</div>
```

De kaart heeft wat CSS voor de breedte en algemene opmaak:

```
.card {
  width: 600px;
  border: 1px solid #171717;
  border-radius: 8px;
}
```



Afbeelding 10.69 Het resultaat met de normale flow (10_71.html).

In de normale flow staan de afbeelding en de tekst onder elkaar. Om ze naast elkaar te krijgen wordt van de kaart een flexcontainer gemaakt. Tegelijk wordt de uitlijning op de kruisjes ingesteld op flex-start. Dat is nodig om te voorkomen dat de standaardinstelling stretch de afbeelding oplekt tot de hoogte van de container.

De alinea <p> wordt ingesteld op groeien met flex: 1. De afbeelding houdt de standaardinstelling flex: 0 1 auto en groeit daarom niet. Zo kan de tekst de vrije ruimte opvullen.

```
.flex {
  display: flex;
  align-items: flex-start;
}

p {
  flex: 1;
  margin: 0;
  padding: 8px;
}
```

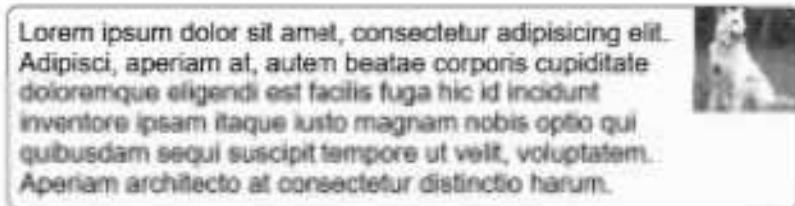


Lorem ipsum dolor sit amet, consectetur adipisicing elit.
 Adipisci, aperiam at, autem beatae corporis cupiditate
 doloremque eligendi est facilis fuga hic id incident
 inventore ipsam itaque iusto magnam nobis optio qui
 quibusdam sequi suscipit tempore ut velit, voluptatem.
 Aperiam architecto at consectetur distinctio harum.

Afbeelding 10.70 Afbeelding aan de linkerkant (10_71.html).

Om de afbeelding naar rechts te verplaatsen is maar één regel CSS nodig die wordt toegevoegd aan de flexcontainer:

```
.flex {
  flex-flow: row-reverse;
}
```



Afbeelding 10.71 Afbeelding aan de rechterkant (10_71.html).

Toepassing: nos.nl

Bij het nabouwen van de site nos.nl met gridlayout is ook flexbox gebruikt (zie de paragraaf *Toepassing nos.nl*). Het gaat om deze HTML (10_54.html):

```
<section class="breaking">
  <div class="breaking-item">
    <h2>Breaking item 1 met een kop over twee regels</h2>
  </div>
  <div class="breaking-item">
    <h2>Breaking item 2 met een kop die ook wat langer is</h2>
  </div>
</section>
```



Afbeelding 10.72 Dit is het gewenste resultaat. De koppen in de sectie *breaking* moeten binnen het centrale blok van maximaal 1200px breed blijven.

De *.breaking-items* overspannen wel altijd de hele breedte van het venster, maar de koptekst moet links en rechts binnen het centrale blok van 1200px blijven. Dat is gerealiseerd met de volgende CSS:

```
.breaking {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(360px, 1fr));
}

.breaking-item {
  display: flex;
  justify-content: flex-start;
  align-items: flex-end;
}

@media screen and (min-width: 1200px) {
  .breaking-item:nth-child(odd) {
    justify-content: flex-end;
  }
}
```

```
.breaking-item h2 {
  max-width: calc(1rem + 600px);
  padding: 0 1rem;
}
```

Analyse van de code

Het `.breaking-item` is een flexcontainer waarin de items in de rijrichting worden geplaatst. De hoofdrichting is dus van links naar rechts en de kruisrichting van boven naar beneden.

De kop `<h2>` is een flexitem dat is links en tegen de onderkant is uitgelijnd met `justify-content: flex-start` en `align-items: flex-end`. Tot 1200px vensterbreedte staat de kop goed, maar als het venster breder wordt, zou de linkerkop tegen de linkervensterrand blijven staan (probleem 1). De rechterkop staat op de goede plek, maar wordt te breed, omdat een kop nu eenmaal een blokbox maakt die met het venster mee groeit (probleem 2).

Probleem 2

Probleem 2 moet eerst worden opgelost. De oplossing bestaat uit het beperken van de breedte van de box tot 600px (de helft van 1200) plus 1rem. Die 1rem komt van de padding die de kop op een smal scherm nodig heeft om vrij te blijven van de vensterrand. Omdat CSS kan rekenen, past deze oplossing in een formule (de functie `calc()` wordt besproken in hoofdstuk 13):

```
.breaking-item h2 {
  max-width: calc(1rem + 600px);
  padding: 0 1rem;
}
```

Probleem 1

Probleem 1 kan nu eenvoudig worden opgelost door de uitlijning van de linkerkop te veranderen van `flex-start` naar `flex-end`. Vanaf 1200px wordt dat in een media query ingesteld voor elke oneven kop, want de oneven koppen staan aan de linkerkant. Hier zijn het maar twee koppen, maar het zouden er ook meer kunnen zijn.

```
media screen and (min-width: 1200px) {
  .breaking-item:nth-child(odd) {
    justify-content: flex-end;
  }
}
```

Samenvatting

In dit omvangrijke hoofdstuk zijn alle lay-outtechnieken van CSS besproken. Met deze kennis kunt u met HTML en CSS alles bouwen wat u maar kunt bedenken.

- Websites met een vast formaat zijn ondenkbaar in het mobiele tijdperk. Responsive design is een techniek om sites zo te bouwen dat ze op elk schermformaat bruikbaar zijn.
- Media queries zijn een belangrijk hulpmiddel bij responsive design. Met een media query kan de opmaak van een element worden veranderd op basis van eigenschappen van de viewport: de breedte of hoogte, portret of landschap, resolutie enzovoort.
- Een tweede hulpmiddel voor responsive design is de metatag `viewport`, waarmee een webpagina zonder schaling op een smartphone of tablet kan worden getoond.
- De basis van elke lay-out is de normale flow: de weergave van de elementen wordt daarbij bepaald door het boxmodel, het weergavemodel en de volgorde in de HTML.
- Met positionering worden elementen op een andere plek gezet; relatief ten opzichte van de oorspronkelijke plek, absoluut zonder dat er een oorspronkelijk plek is. De verplaatsing is altijd ten opzichte van de container van het element. Om absoluut te kunnen positioneren moet die container relatief zijn gepositioneerd.
- Met `fixed` wordt een element op een schermpositie vastgezet; het element scrollt niet. Met `sticky` wordt een element ook vastgezet, maar scrollt het mee met de onderkant van de container.
- Een float is een blokbox die uit de normale flow wordt gehaald en naar links of rechts verschuift. Tekst kan om een float heenlopen en dat maakt een float handig voor afbeeldingen in een tekst.
- Floats waren vanwege hun flexibiliteit tot de komst van gridlay-out en flexbox een belangrijk techniek voor de lay-out van pagina's. Daar zijn ze nu niet meer voor nodig.
- Een block formatting context (BFC) is een soort privéterrein binnen de normale flow, met een hek om de inhoud. Een element dat een BFC maakt, omvat alle elementen die erin zitten; het voorkomt dat floats buiten de container kunnen staan. Het voorkomt ook het samenvoegen van marges. Een box kan een BFC worden door `display` de waarde `block-root` te geven.
- Multi-column is een lay-outtechniek waarmee inhoud wordt geplaatst in een vast aantal kolommen, in kolommen met een bepaalde breedte of een combinatie daarvan. De techniek heeft beperkingen, maar is bijvoorbeeld

geschikt voor een menu met een vast formaat of een reeks kaarten met verschillende hoogten.

- Gridlay-out is de laatste stand van de lay-outtechniek in CSS. Er wordt een raster van horizontale en verticale lijnen mee gemaakt waarop onderdelen van de pagina worden geplaatst. Grid is een tweedimensionaal systeem: items worden tegelijk in rijen en kolommen neergezet.
- Het raster bestaat uit tracks. Een track is het gebied tussen twee rasterlijnen.
- Items plaatsen kan met lijnnummers, lijnen met een naam en gebieden met een naam.
- De tracks kunnen vaste afmetingen hebben, flexibel zijn of een combinatie daarvan.
- Hulpmiddelen bij te maken van flexibele tracks zijn repeat(), auto-fit, auto-fit, minmax() en de eenheid fr.
- Een raster is nooit te klein; als er meer items dan tracks zijn, worden tracks bijgemaakt. Voor automatisch gemaakte tracks kan een afmeting worden ingesteld.
- Items worden uitgelijnd en de rijrichting en de kolomrichting. Ook afzonderlijke items kunnen worden uitgelijnd en het hele raster kan in zijn container worden uitgelijnd. Het verticaal centreren van boxen is geen enkel probleem meer.
- Ook Flexbox is een relatief nieuwe lay-outtechniek. Met flexbox zijn items in één richting flexibel. Dat is gelijk het grote verschil met gridlay-out, dat in twee richtingen werkt.
- Flexbox is vooral geschikt voor het bouwen van componenten en minder handig voor paginalay-outs.
- Met flexbox kunnen items groeien en krimpen in verschillende verhoudingen ten opzichte van elkaar.
- Flexitems worden uitgelijnd op de hoofdas, dat is de flexibele richting, en op de kruisdas, die haaks op de hoofdas staat. Ook met flexbox is verticaal centreren van boxen geen probleem.

Oefeningen

- Experimenteer met verschillende typen positionering:
 - Relatieve positionering.
 - Absolute positionering op de pagina, maar ook in een blok. Hoe zorgt u ervoor dat een willekeurig blok geschikt is om een element absoluut te positioneren?
 - Maak een blokje dat een vaste positie heeft halverwege aan de zijkant van de pagina, bijvoorbeeld met daarin de tekst *Feedback*. Zoek uit hoe zo'n tekst rechtop kan worden gezet.
 - Maak een header die aan de bovenkant van de pagina blijft staan terwijl andere inhoud eronderdoor schuift.

- Laat een afbeelding in een tekst met float een nieuwe positie innemen terwijl de overige tekst eromheen loopt.
- Maak een artikel en plaats daarin diverse gefloateerde afbeeldingen. Gebruik daarbij ook alle toepasselijke elementen die u kent, waaronder <article>, <figure>, <figcaption>.
- (Voer deze oefening uit in Chrome.) Maak een artikel en laat de tekst met behulp van multi-column over meerdere kolommen lopen.
 - Maak de kolommen flexibel door een aantal en een breedte in te stellen.
 - Laat de kop alle kolommen overspannen.
 - Voeg een afbeelding met bijwerkingsvoegstuk toe en zorg ervoor dat die bij elkaar blijven.
- Open de pagina belastingdienst.nl en bekijk de code van de header met de Inspector in de browser (Ctrl+Shift+I). Het gaat vooral om de hoeveelheid code die gebruikt wordt om het logo midden boven aan de pagina te plaatsen:

```
<header>
<section class="Logo">
  <div class="container">
    <div class="row">
      <div class="col-xs-12">
        <a href="">
          <img src="" alt="Logo">
        </a>
      </div>
    </div>
  </div>
</section>
<nav> ... </nav>
</header>
```

Dit is typisch code van een twaalfkoloms raster uit UI-framework, waarbij in een container een row wordt gemaakt met daarin een kolomcontainer die vanaf de kleinste weergave (xs) twaalf kolommen overspant (col-xs-12); dat is dus bij elke schermbreedte:

- Herschrijf de code van de logosectie. Verklein het aantal 'lege' elementen (div) en kies de handigste methode om een logo te centreren bij elke venstergrootte.
- Is <section> het juiste element op deze plek? Beargumenteer uw antwoord.
- Bouw een kaartcomponent, bijvoorbeeld zoals in afbeelding 10.67. Het verschil moet zijn dat de afbeelding aan de bovenkant staat en dat de kop op de afbeelding staat (eventueel op een achtergrondkleur voor het contrast).

- Probeer het met verschillende technieken: absoluut positioneren, float, gridlay-out. Kan dit ook met flexbox?
- Bouw de indeling van de component Plan uw reis van ns.nl na (alleen de lay-out). Maak ook uw versie responsive. Bekijk van het origineel de weergave van klein naar groot in de Inspector met de responsive-designmodus (Ctrl+Shift+M).
 - Begin met het antwoord op de vraag in hoeveel richtingen het ontwerp flexibel moet zijn. Bepaal dan of u gridlay-out of flexbox gebruikt; of combinatie van deze en andere technieken.
- Bouw met gridlay-out een complete paginalay-out. Het gaat niet om de opmaak (daar gaan de volgende hoofdstukken over), maar om de indeling. Het hoeft geen origineel ontwerp te zijn, de indeling van bestaande pagina's namaken is een heel goede oefening.
 - Maak de pagina ook responsive, mobile first.
 - Maak van een pagina een schets op papier met de onderdelen die een plek moeten krijgen. Beslis op basis van de schets welke lay-outtechnieken nodig zijn.
 - Kijk bijvoorbeeld eens naar nos.nl, nu.nl of developer.mozilla.org/nl/.



Gebruik de voorbeeldcode

Van vrijwel alle afbeeldingen is de HTML- en CSS-code beschikbaar. De bestandsnaam staat in het bijschrift. Gebruik deze code om te oefenen: pas de code aan, voeg dingen toe; speel ermee! Kijk op handboek-html-css.nl.

Tekst en typografie

Dit hoofdstuk gaat dieper in op de eigenschappen en waarden voor tekst en typografie. Het begint met het instellen van een lettertype, dat net zo makkelijk van internet als van het systeem kan worden gehaald. Is eenmaal een lettertype ingesteld, dan kunnen daarvan allerlei eigenschappen worden aangepast: grootte, zwaarte, regelhoogte enzovoort. Ook andere aanpassingen zijn mogelijk: afstand tussen letters en woorden, witruimte, uitlijning en inspringen.

U leert in dit hoofdstuk:

Verschillende lettertypen gebruiken.

Lettertypen downloaden met @font-face.

Eigenschappen voor lettertypen instellen.

Werken met teksteigenschappen zoals inspringen en uitlijnen.

Tekst met schaduweffecten maken.

De opmaak van lijsten aanpassen, ook voor navigatie.

Inleiding

In dit hoofdstuk wordt het gebruik van lettertypen en de vormgeving van tekst beschreven. De eigenschappen komen deels uit CSS 2.1 en verder uit diverse CSS-modules, waaronder:

- *CSS Fonts level 3*
- *CSS Text level 3*
- *CSS Text Decoration level 3*



Alle eigenschappen

Een handig overzicht van alle CSS-eigenschappen is te vinden op developer.mozilla.org/en-US/docs/Web/CSS/Reference/. Elke eigenschap is gekoppeld aan uitleg met een link naar de bijbehorende CSS-module. Zo hoeft u niet allemaal verschillende CSS-modules te doorzoeken.

De CSS-eigenschappen waar het in dit hoofdstuk om gaat, hebben betrekking op de te gebruiken lettertypefamilies, maar ook op de grootte, de stijl (cursief) en de zwaarte (vetheid) van het lettertype. Er zijn ook meer algemene teksteigenschappen en dan gaat het om witruimte, uitlijning, inspringen en omzetting naar hoofdletters. Uiteraard begint het allemaal met het kiezen van een lettertype en het beantwoorden van de vraag: waar haalt u dat vandaan?

Lettertype: van systeem of online?

Om de open deur nog maar eens een trap na te geven: tekst is nog steeds voor het gros van de websites het belangrijkste ingrediënt – ondanks veel (aanname) visueel geweld. Dat die tekst de lezer natuurlijk boeit en in zijn informatiebehoefte voorziet, is net zo'n open deur. Hoe die tekst eruitziet krijgt echter vaak minder aandacht, terwijl een goed gekozen lettertype met een 'juiste' weergave voor een belangrijk deel de uitstraling van de pagina bepaalt. Nu is 'juist' een nogal vaag begrip waarvan de betekenis van geval tot geval verschilt, maar waarschijnlijk begint u zelf ook niet aan een pagina met prieelige tekst of een woest fantasielettertype. Hoe dan ook, we laten de keus van de lettertypen op uw site graag aan uw goede smaak over. Hier gaat het om de vraag waar u lettertypen vandaan haalt en hoe u ze op uw site krijgt.

Jarenlang is gebruikgemaakt van de lettertypen die tegelijk met het besturingssysteem op de computer van de ontwerper waren geïnstalleerd. Niet omdat dit de beste optie was; het was de enige optie. Het had namelijk weinig zin om zelf een ander lettertype te installeren en voor de website te gebruiken als niet

zeker was de bezoeker hetzelfde lettertype had geïnstalleerd. Tegenwoordig bevatten besturingssystemen meer lettertypen en is de keus wat groter, maar het blijft behelpen.

Web fonts

Er is echter een nieuwe bron van lettertypen beschikbaar gekomen: internet. Hoewel, nieuw. Het principe om gebruik te kunnen maken van gedownloade lettertypen bestaat al lang, maar was in de praktijk nutteloos omdat niet alle belangrijke browsers ermee overweg konden en doordat de manier waarop het werkte per browser verschildde. In de nieuwe generatie webbrowsers is het wel op een gestandaardiseerde manier ingebouwd en prima bruikbaar. Dat is goed nieuws, want nu kunt u zich onderscheiden met goed gekozen lettertypen die niet al op elke website te zien zijn. Of juist kiezen voor een zeer vertrouwd lettertype, dat in elke site of app gebaseerd op het ontwerpsysteem van Google wordt gebruikt. (Dat is Roboto.) Ook een ander probleem dat het gebruik van gedownloade lettertypen afremde, copyright en betaling voor het gebruik ervan, is inmiddels opgelost.

Voor weblettertypen zijn verschillende bestandsformaten beschikbaar, maar voor moderne browsers hebt u er maar één nodig: WOFF of WOFF2. WOFF staat voor Web Open Font Format en de specificatie is sinds 2012 een W3C-aanbeveling (*WOFF File Format 1.0*). WOFF2 (*WOFF File Format 2.0*, zie www.w3.org/TR/WOFF2/) is een doorontwikkeling van WOFF met een betere compressie. WOFF is niet een lettertypeformaat zoals True Type of Open Type (TTF en OTF). Het is een container voor lettertypen met een opbouw zoals TTF en OTF, maar de informatie wordt gecomprimeerd en aangevuld met metadata-gevens over bijvoorbeeld toegestaan gebruik. Vooral dat comprimeren is belangrijk om de hoeveelheid te transporteren data en daarmee de download zo klein mogelijk te houden. (Van WOFF2 wordt gezegd dat het 30 tot soms 50% betere compressie geeft dan WOFF.) Na het downloaden pakt de browser het lettertype uit om het te kunnen gebruiken in de website. Het lettertype wordt niet geïnstalleerd op het apparaat van de bezoeker. Zo wordt voorkomen dat commerciële lettertypen zonder betaling worden doorgegeven.

De volgende paragraaf gaat over de CSS-regel waarmee lettertypen worden gedownload en voorbereid op gebruik. U hebt die informatie niet nodig wanneer u de standaard systeemlettertypen gebruikt die op elk apparaat beschikbaar zijn. Blader dan door naar de paragraaf *Eigenschappen voor lettertype: font*. Het gebruik van weblettertypen is echter zo algemeen, dat we adviseren om deze informatie niet over te slaan.

The screenshot shows the Google Fonts homepage. At the top, there are navigation links: DIRECTORY, FEATURED, ABOUT, and a search bar labeled 'Search'. Below the navigation, there's a section titled 'Viewing 815 of 815 font families'.

Two font families are displayed side-by-side:

- Roboto** by Christian Robertson (12 styles): The sample text reads, "All their equipment and instruments are alive."
- Spectral** by Production Type (14 styles): The sample text reads, "A red flair silhouetted the jagged edge of a wing."

Below these, another pair of fonts is shown:

- Joelín Sans** by Santiago Orozco (10 styles): The sample text reads, "I watched the storm, so beautiful yet terrific."
- Open Sans** by Steve Matteson (10 styles): The sample text reads, "Almost before we knew it, we had left the ground."

On the right side of the page, there are several filtering options:

- Categories:** Includes boxes for Serif, Sans-Serif, Display, Handwriting, and Monospace.
- Sorting:** A dropdown menu set to 'Trending'.
- Languages:** A dropdown menu set to 'All Languages'.
- Number of styles:** A slider from 0 to 100.
- Thickness:** A slider from 0 to 100.
- Slab:** A slider from 0 to 100.
- Width:** A slider from 0 to 100.

Afbeelding 11.1 Bij Google Fonts vindt u gratis lettertypen.

The screenshot shows the Adobe Fonts website. At the top, there's a search bar with placeholder text 'Search fonts, families, designers, and more...' and a 'My Adobe Fonts' link.

The main interface includes:

- Clear Filters** button.
- Font Filter** buttons: New, Hot, and Sale.
- Default** and **Advanced** filter buttons.
- Font Size** slider.
- CATEGORIES** dropdown menu with options: All, Sans-Serif, Serif, Slab-Serif, Script, Handwriting, Mono, Hand, and Decorative.
- DOCUMENTATION** dropdown menu with options: All, Paragraph, and Headings.
- OPTIONS** dropdown menu with options: All, Weight, and Width.
- Font Preview Area:** Shows two examples of the 'Mr Eaves' font family. The first example uses the 'Mr Eaves Sans' weight and the second uses the 'Mr Eaves XL' weight. Both examples show the text 'The quick brown fox jumps over the lazy dog'.
- Font Details:** Below each preview area, it shows the font name, weight, and character count (e.g., 12 fonts).

Afbeelding 11.2 Adobe Fonts is een abonnementsdienst.

Er zijn veel leveranciers van web fonts op internet te vinden. Twee aanbieders noemen we met name:

- Google Fonts, te vinden op www.google.com/fonts/. Hier kunt u gratis gebruikmaken van lettertypen. U krijgt ze niet in bezit, maar verwijst ernaar vanuit uw webpagina. In feite hebt u het lettertype te lenen.
- Adobe Fonts, te vinden op fonts.adobe.com/fonts (voorheen bekend als Typekit). Een groot aanbod van lettertypen, maar alles tegen betaling. De bibliotheek is beschikbaar als onderdeel van de abonnementsservice Adobe Creative Cloud. Hier huurt u lettertypen voor de duur van het abonnement.

Lettertypen downloaden: @font-face

Met de regel @font-face wordt een koppeling gemaakt naar een downloadbaar lettertypebestand op internet. In de praktijk hoeft u @font-face echter lang niet altijd zelf in te stellen. Wanneer u lettertypen 'huurt' zoals bij Adobe Fonts of 'leent' zoals bij Google, krijgt u op basis van uw keuzen een code. Deze code is of een <link> naar een stylesheet op de server van de aanbieder of een JavaScript-code, die u in uw eigen HTML-bestanden opneemt. Daarmee wordt een door de externe server gegenereerd CSS-bestand geladen, dat de lettertypeverwijzing aan uw site toevoegt. U installeert dus niet zelf de lettertypen en kopieert ze ook niet naar uw webserver, en schrijft zelfs niet de bijhorende @font-face-declaraties zelf. Die staan in het CSS-bestand dat van de externe server komt. Op deze manier houden copyrighthouders controle over hun lettertypen en u hoeft zich niet te bekommeren om de @font-face-regel.



Bekijk de externe CSS-code

Wanneer u bijvoorbeeld een Google-font gebruikt en een webpagina opent waarin de code staat, kunt u met de Hulpmiddelen voor ontwikkelaars de CSS-code bekijken die Google aan uw pagina heeft toegevoegd om het lettertype beschikbaar te maken. In Chrome/Canary kiest u F12, optie Sources, tab Sources, font.googleapis.com. Klik op het CSS-bestand dat daar staat om de inhoud ervan te bekijken.

Koopt u lettertypen, dan beschikt u zelf over de fysieke bestanden (bij voorkeur ook van het type WOFF/WOFF2 vanwege de brede browserondersteuning). Om ze op uw site te kunnen gebruiken, moeten de lettertypebestanden op de webserver worden gezet. Vervolgens gebruikt u de regel @font-face om het lettertype beschikbaar te maken voor uw bezoekers.



Omzetten naar WOFF

Beschikt u over TTF- of OTF-lettertypen met een licentie, dan kunt u deze via online tools laten omzetten naar WOFF/WOFF2. Bijvoorbeeld Font Squirrel biedt een *webfont generator* aan: www.fontsquirrel.com/tools/webfont-generator.

Eigenschappen van @font-face

Deze paragraaf geldt alleen voor lettertypen die u zelf host. Als u gebruikmaakt van bijvoorbeeld Google Fonts leest u de paragraaf *Een embedcode gebruiken*.

De regel `@font-face` komt of als eerste in het CSS-stijlblad of in een andere `@`-regel (bijvoorbeeld `@media`, om op mobiele apparaten een ander lettertype te gebruiken dan in een desktopbrowser). Er mogen meerdere `@font-face`-regels worden gebruikt, zodat de mogelijkheid bestaat om voor de site een lettertypebibliotheek aan te leggen. De basis van de regel bestaat uit de naam van het lettertype en de bronlocatie (de server van de leverancier of een map op uw webserver):

```
@font-face {  
    font-family: naam-lettertype;  
    src: url('naam-lettertype.woff2');  
}
```



Let op de locatie van de map

Biedt u lettertypen aan vanaf uw eigen server, bedenk dan dat u de locatie van de map met lettertypen moet zien vanuit de map met CSS-bestanden. Dat is gewoonlijk een submap van de webroot. Als uw lettertypen ook in een submap van de webroot staan, moet u dus eerst omhoog naar de root en dan omlaag naar de lettertypen (zie ook hoofdstuk 5).

De *naam-lettertype* achter `font-family` is wat u eraan geeft. Deze regel is een toewijzing van een naam aan een lettertypebestand. Het is in feite een variabele. Hoewel het gebruikelijk is om hier de naam van het lettertype in te vullen, mag het ook een beschrijvende naam zijn, zoals `bodytekst` of `koptekst`. Dit kan overigens alleen als u zelf het lettertype op de webserver hebt staan, want anders kunt u `font-family` niet aanpassen.



CSS-variabelen

Sinds 2017 kunnen in CSS variabelen worden gebruikt, waarin bijvoorbeeld het lettertype voor de bodytekst en koppen kan worden opgeslagen. CSS-variabelen (*CSS Custom Properties*) komen aan bod in hoofdstuk 13.

Staan er spaties in de lettertypenaam, dan moet deze waarde bij `font-family` tussen aanhalingstekens worden geplaatst. Vergelijk:

```
font-family: Helvetica;
font-family: 'Times New Roman';
```

De varianten van een lettertype kunnen met verschillende @-regels worden opgehaald. Het voordeel daarvan is dat u niet per se een complete lettertype-familie hoeft te downloaden, want die kan heel omvangrijk zijn, tot vele megabytes. Met de volgende regels worden van een lettertype alleen de normale, vette en cursieve versie gebruikt. Met `font-weight` of `font-style` wordt respectievelijk vet en cursief aangegeven.



Let op vet en cursief

Download altijd maast de gewone variant een vette en een cursieve variant van een lettertype, aangenomen dat u ook vette en cursieve tekst wilt gebruiken. Een browser kan weliswaar vet of cursief simuleren, maar dat zal nooit meer zijn dan een benadering. Bij gewone lettertypen kan het resultaat acceptabel zijn, maar bij decoratieve lettertype meestal niet. Voor écht vet of écht cursief moeten de juiste lettertypevarianten beschikbaar zijn.

```
@font-face {
    font-family: SourceSansPro;
    src: url('fonts/SourceSansPro-Regular.woff2');
}

@font-face {
    font-family: SourceSansPro;
    src: url('fonts/SourceSansPro-Bold.woff2');
    font-weight: bold;
}

@font-face {
    font-family: SourceSansPro;
    src: url('fonts/SourceSansPro-It.woff2');
    font-style: italic;
}
```

```
/* voorbeelden van CSS-regels om het lettertype te gebruiken */  
p { font-family: SourceSansPro; }  
p { font-family: SourceSansPro; font-weight: bold; }  
p { font-family: SourceSansPro; font-style: italic; }
```

Het attribuut format

Bij het downloaden kan worden voorkomen dat de browser een lettertype laadt in een formaat dat deze niet ondersteunt door bij src ook het attribuut format te gebruiken. Een niet-ondersteund formaat wordt niet gedownload:

```
@font-face {  
    font-family: SourceSansPro;  
    src: url('fonts/SourceSansPro-Regular.woff2') format('woff2');  
}
```

Met de goede ondersteuning van WOFF2 door moderne browsers is dit bijna overbodig, maar als u ook oudere browsers wilt bedienen, kunt u met format een extra lettertypeformaat beschikbaar stellen, namelijk TTF. De @-regel wordt dan:

```
@font-face {  
    font-family: SourceSansPro;  
    src: url('fonts/SourceSansPro-Regular.woff2') format('woff2');  
    src: url('fonts/SourceSansPro-Regular.woff') format('woff');  
    src: url('fonts/SourceSansPro-Regular.ttf') format('truetype');  
}
```

Het attribuut local

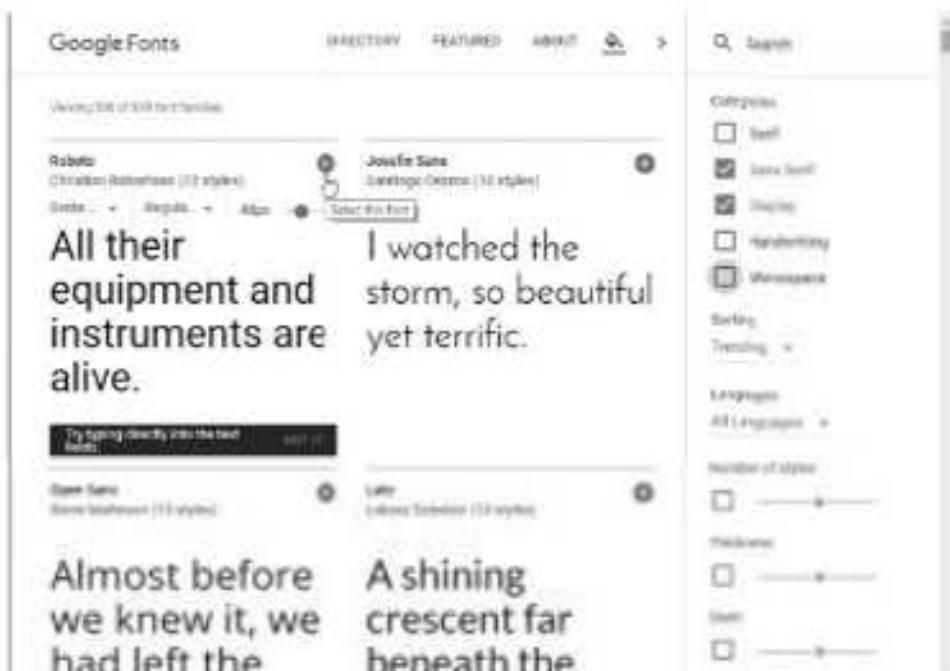
Met het attribuut local in de eigenschap src kan de lokale versie van een lettertype worden gebruikt, uiteraard mits dit op het systeem van de bezoeker staat. Dat scheelt het downloaden ervan, maar de kans bestaat (klein, maar toch) dat achter de naam een ander lettertype schuilgaat dan u denkt. Omdat u niet weet met welke schrijfwijze de lettertypenaam op het systeem van de gebruiker staat, biedt u varianten aan:

```
@font-face {  
    font-family: SourceSansPro;  
    src: local('Source Sans Pro'),  
        local('Source-Sans-Pro'),  
        url('fonts/SourceSansPro-Regular.woff2') format('woff2'),  
        url('fonts/SourceSansPro-Regular.woff') format('woff');  
}
```

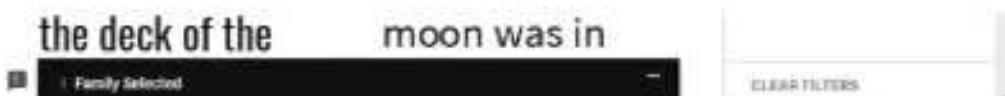
Een embedcode gebruiken

Voor lettertypen die u niet als fysiek bestand op uw computer hebt staan, krijgt u van de leverancier een embedcode. Let erop dat deze code in elke pagina moet staan waarop het desbetreffende lettertype wordt gebruikt (of als @import in het stijlblad dat u vanuit elke pagina aanroept). De bekendste leverancier van gratis lettertypen is Google. We laten in vogelvlucht zien hoe u hiervan een gratis weblettertype in uw pagina gebruikt.

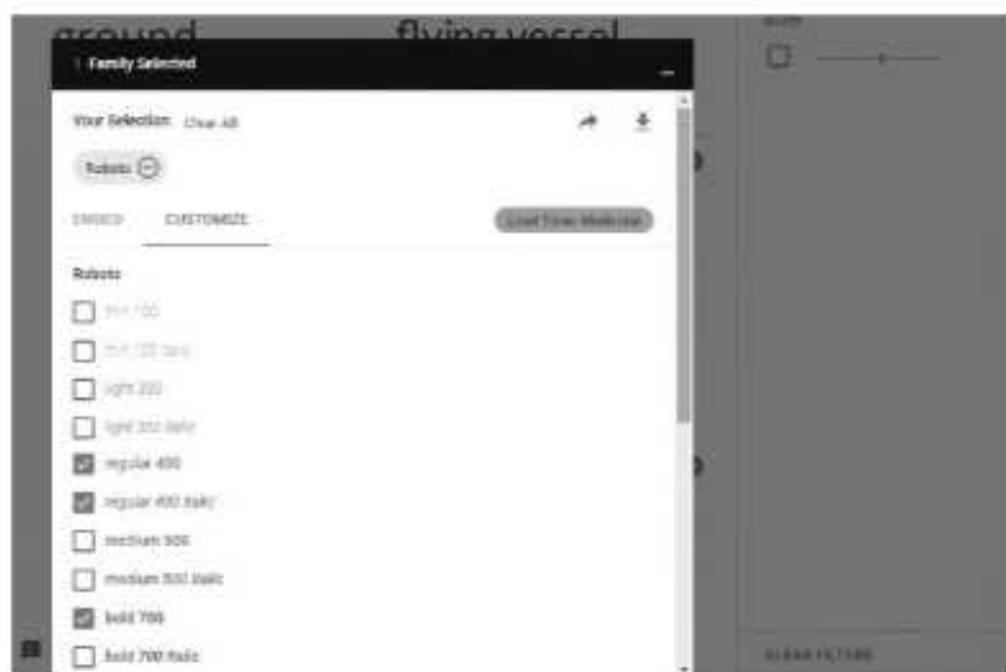
- 1 Ga naar www.google.com/fonts. Hier kunt u kiezen uit ruim 900 lettertypen.
- 2 Verklein de selectie met de filters aan de rechterkant. Kies bijvoorbeeld alleen de categorieën Sans Serif en Display voor een schreefloze schermletter.
 - Het deelvenster links toont de beschikbare lettertypen in verschillende weergaven.
- 3 Klik bij het gewenste lettertype op de knop Select this font (een plussteken).
 - Het plussteken is een min (Deselect this font) als u het lettertype eerder hebt toegevoegd.
- 4 Klik onder in het venster op het de selectielade (1 Family Selected).
- 5 Kies de gewenste stijlen. Houd de indicator Load Time in de gaten. Kies niet meer dan u echt nodig hebt.
Kies zo nodig aanvullende talen.
- 6 Klik op Embed en kopieer de <link>-code.
- 7 Plak de <link>-code in de <head> van de HTML-pagina;
 - of plak de @import-regel in een <style>-element in de <head>;
 - of plak de korte @import-regel in een stijlblad (zonder <style>-element).



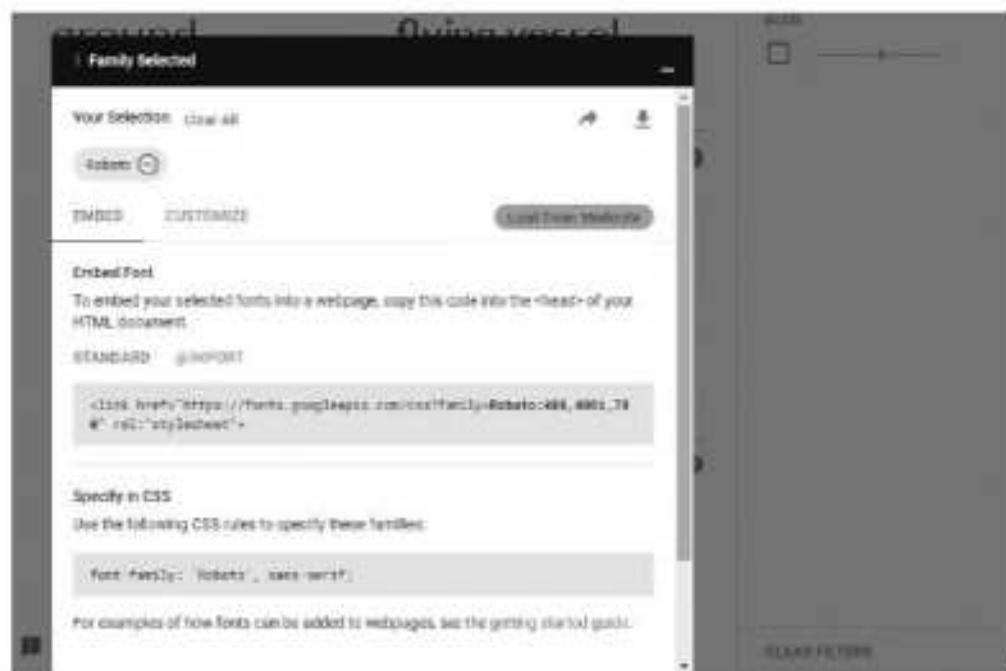
Afbeelding 11.3 Filter de selectie op schreefloze en schermlettertypen, kies een lettertype en klik op het plussteken.



Afbeelding 11.4 De lade onder aan de pagina bevat één lettertype. Klik erop om door te gaan.



Afbeelding 11.5 Selecteer de gewenste lettertypestijlen en houd de Load Time in de gaten.



Afbeelding 11.6 Kopieer de code en lees de gebruiksaanwijzing.

Eigenschappen voor lettertype: font

Uit welke bron de lettertypen op de website ook komen, de instellingen maakt u met de eigenschappen van font:

- font-style
- font-variant
- font-weight
- font-size
- line-height
- font-family

Deze volgorde is niet willekeurig gekozen. Als u de waarden opneemt in de verkeerde eigenschap font, moeten ze in deze volgorde staan. Verplicht bij het gebruik van font zijn alleen font-size en font-family. Een weggelaten eigenschap wordt ingesteld op de initiële waarde.

Lettertype

Het lettertype van elementen wordt aangegeven met de eigenschap font-family. U kunt in theorie elk lettertype kiezen, maar als u geen weblettertype hebt ingesteld en het genoemde lettertype staat niet op de computer van de bezoeker, dan kiest de browser zelf een algemeen lettertype, bijvoorbeeld Arial, Helvetica, Times New Roman of Courier New. Een voorbeeld is:

```
font-family: Segoe, "Segoe UI", "DejaVu Sans", "Trebuchet MS", Verdana, sans-serif;
```

Dit is een combinatie van lettertypen die zo veel mogelijk weergaveapparaten bestrijkt en sans-serif is een algemene aanduiding voor een schreefloos lettertype. De browser verwerkt de regel van links naar rechts en controleert of het lettertype beschikbaar is op het apparaat van de gebruiker. Wordt geen overeenkomst met een benoemd lettertype gevonden, dan wordt een algemeen lettertype gebruikt. Vandaar dat wordt aanbevolen altijd de lijst altijd te eindigen met een generieke lettertypeaanduiding: sans-serif (schreefloos), serif (met schreef), cursive (cursief), fantasy (niet voor gewone tekst) of monospace (schermttekst). Welk lettertype de browser in dat geval gebruikt is niet omschreven, maar op een Windows-desktop wordt het meestal Arial, Times New Roman of Courier New/Consolas.

Lettertypenamen zijn hoofdlettergevoelig en als de naam uit meer woorden bestaat, plaatst u er aanhalingstekens omheen, zoals in het codevoorbeeld dat hierboven staat.

Zo veel lettertypen

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Provident maiores, veniam veritatis quis! Nisi officia vel ratione iure eveniet vitae id. Voluptatum neque, fuga optio sint mollitia itaque asperiores a.

Quas doloremque tempora. Vero molestiae reprehenderit, doloribus. Ut debitis, ducimus corrupti, delectus perferendis odio et at rem iusto labore consequatur.

Autem quasi, quidem rerum ducimus possimus, hic sunt, quo distinctio aperiam fugit quis odio minus porro nemo, dolor quam ea mollitia recusandae.

Id voluptate non at, neque natus eius ad facilis dolor. Ea, reiciendis incident aliquid doloribus quaerat vero ipsa repudiandae vitae a repellat.

Afbeelding 11.7 Verschillende lettertypen ingesteld met CSS.

Lettergrootte: waarden en overwegingen

De grootte van het lettertype wordt ingesteld met de eigenschap `font-size`. Het formaat van de tekst kan op verschillende manieren worden aangegeven:

- absoluut xx-small, x-small, small, medium, large, x-large, xx-large;
- relatief larger of smaller;
- een lengte een getal in punten, pixels, em, rem, vw, vh, maar ook ch, ex en meer;
- percentage een getal in procenten.

Over welke eenheid u 'moet' gebruiken voor tekst wordt al zo lang gediscussieerd als CSS bestaat, net als over de grootte zelf trouwens. En dat is niet minder geworden met modern responsive design en de mogelijkheid om viewporteenheden te gebruiken voor tekstgrootte.

Belangrijk is dat u weet dat de basissinstelling van tekst in een browser 16px is en dat die grootte is ingesteld op het rootelement `<html>`. Alle andere elementen erven die grootte en de kopteksten blijven in verhouding, want die hebben in de browserstylesheet standaard al een grootte relatief aan de basistekst. Ook belangrijk is dat u rekening houdt met de toegankelijkheid van uw tekst en dat gebruikers die kunnen lezen zonder dat ze hoofdpijn krijgen van het ingespannen turen naar veel te kleine lettertjes.

Minstens zo belangrijk is dat u consequent bent en ervoor zorgt dat gelijksoortige tekst even groot is en dat alle koppen onderling een logische variatie in

grootte hebben. Richt bijvoorbeeld in uw stylesheet een sectie in waarin u het formaat van alle typen tekst in uw site definieert.

Op kleine schermen kunt u wat kleinere tekst gebruiken. De kijkafstand is kleiner en de regellengte blijft enigszins in de buurt van wat comfortabel leest; ideaal tussen 45 tot 75/85 tekens inclusief spaties en leestekens. Pas op dat tekst niet te klein wordt in uw poging om de regellengte op peil te houden. Een vuistregel is 16 pixels (de standaardgrootte), maar elk lettertype is anders.

Als 16px de leidraad is voor mobiele schermen, begrijpt u dat op tablets, laptops en desktops grotere tekst gewenst is. 20 tot 24 pixels – ook weer afhankelijk van het lettertype en de aard van de tekst – is niet bijzonder. Hier is het nog belangrijker dat u de regellengte niet laat oplopen tot boven de 85 tekens. Verder is een vuistregel voor de regelhoogte (`line-height`) 1,5 maal die tekstgrootte; maar ook dat is geen wet.



Eenheden in hoofdstuk 8

In hoofdstuk 8, de paragraaf *Waarden en eenheden*, leest u alles over px, em, rem en meer.

Geavanceerd: responsieve tekst

Nu responsive design de standaard in webdevelopment is geworden, wordt ook meer nagedacht over de toepassing van vloeientechniek. Vooral met moderne viewporteenheden lijkt vloeientechniek beter mogelijk. Helaas is het nog lang niet eenvoudig om dat op een bruikbare manier voor elkaar te krijgen. Er mee experimenteren is sowieso leuk en er zijn online prima artikelen over te vinden, bijvoorbeeld www.smashingmagazine.com/2016/05/fluid-typography/.

Nog even kort: de viewporteenheden zijn vw (breedte), vh (hoogte) vmin (kleinste van de twee) vmax (grootste van de twee). Zodra font-size wordt gekoppeld aan vw groeit de tekst met het toenemen van de breedte van de viewport.

Op de kleinste mobiel (360px) moet Times New Roman toch wel 16px zijn: 1vw = 3,6, dus 16px is bij 360px ongeveer 4,5vw. Bij een viewport van 600px is de tekst dan echter al 27px en op een iPad (portretstand 768px) al bijna 35px. Dat werkt in elk geval niet.

In het genoemd artikel staat een methode en die is gebruikt voor de volgende afbeeldingen. Die zou met media queries kunnen worden verfijnd en leent zich in elk geval voor experimenteren. We presenteren hier geen kant-en-klare oplossing, maar adviseren u zich in het onderwerp te verdiepen.

Responsive tekst

 Lorem ipsum dolor sit amet, consectetur adipisicing elit. Omnis quibusdam reiciendis ratione dolor animi nostrum quos illo, dignissimos labore praesentium maiores voluptates reprehenderit expedita! Eum voluptate enim maiores ut nisi.

 Maxime odio, voluptates animi? Veniam id nam in qui ratione ullam dolorem tempora a, vero alias, fugiat doloremque officia aperiam distinctio laborum minima possimus! Beatae nisi excepturi possimus, minus explicabo.

 Saepe commodi, asperiores, quasi architecto dolens diligisci open es ratione

Responsive tekst

 Lorem ipsum dolor sit amet, consectetur adipisicing elit. Omnis quibusdam reiciendis ratione dolor animi nostrum quos illo, dignissimos labore praesentium maiores voluptates reprehenderit expedita! Eum voluptate enim maiores ut nisi.

 Maxime odio, voluptates animi? Veniam id nam in qui ratione ullam dolorem tempora a, vero alias, fugiat doloremque officia aperiam distinctio laborum minima possimus! Beatae nisi excepturi possimus, minus explicabo.

 Saepe commodi, asperiores, quasi architecto dolore adipisci

Afbeelding 11.8 Vloeiente teksgrootte op basis van schermbreedten 320px en 600px.

Responsive tekst

 Lorem ipsum dolor sit amet, consectetur adipisicing elit. Omnis quibusdam reiciendis ratione dolor animi nostrum quos illo, dignissimos labore praesentium maiores voluptates reprehenderit expedita! Eum voluptate enim maiores ut nisi.

 Maxime odio, voluptates animi? Veniam id nam in qui ratione ullam dolorem tempora a, vero alias, fugiat doloremque officia aperiam distinctio laborum minima possimus! Beatae nisi excepturi possimus, minus explicabo.

Afbeelding 11.9 Dezelfde berekening toegepast bij een schermbreedte van 800px.

Toepassing van font-size

Gebruikelijke waarden om de teksgrootte te definiëren zijn px, em en rem. Relatieve waarden hebben de voorkeur boven de absolute pixel. Zoals gezegd, gewoonlijk is de standaardgrootte van tekst 16px, en dat is ook de waarde van 1rem. Op een groter scherm mag de tekst wat groter, bijvoorbeeld 20px.

Dat wordt dan bijvoorbeeld:

```
p {
  font-size: 1.25rem;
}
```

Maar het is dan handiger om de instelling voor het hele document aan te passen en daar alle andere grootten van af te leiden:

```
html {
  font-size: 1.25rem;
}
```

Een em erft de grootte van zijn ouder, we nemen aan body, dus alineatekst van 20px is in dit voorbeeld ook:

```
p {
  font-size: 1em;
}
```

Een kop `<h2>` wilt u bijvoorbeeld anderhalf keer zo groot hebben, 30px. Dat kan met em, die nog steeds de waarde 20 heeft:

```
h2 {
  font-size: 1.5em;
}
```

Grootte aanpassen: `font-size-adjust`

De eigenschap `font-size-adjust` is bedoeld om de teksgrootte gelijk te houden wanneer wordt teruggevallen op een ander lettertype in de declaratie van `font-family`. Een lettertype van tweede of derde keus in de declaratie van `font-family` kan kleiner of groter ogen dan het gewenste lettertype (een voorbeeld is het verschil tussen Verdana en Calibri). Met deze eigenschap kan dit verschil worden opgeheven, al vergt dat nogal een berekening. Vanwege de geringe browserondersteuning (in 2019 alleen Firefox) wordt die verder achterwege gelaten. Zie www.w3.org/TR/css3-fonts/#propdef-font-size-adjust voor meer informatie en caniuse.com voor browserondersteuning.

Regelhoogte: `line-height`

Gekoppeld aan de grootte van het lettertype is de eigenschap voor regelhoeveelheid: `line-height`. Hoewel het niet hetzelfde is, kunt u dit beschouwen als de regelafstand, de hoeveelheid wit tussen regels tekst. Elk lettertype heeft een hoeveelheid ingebouwd wit, maar die kunt u groter of kleiner maken om meer of minder 'lucht' in de pagina te brengen.

De standaardwaarde van `line-height` is `normal`. De numerieke waarde van `normal` is ingebouwd in het lettertype. Andere mogelijke waarden zijn:

- een getal: vermenigvuldigd met `font-size` geeft dat de regelhoogte;
- een lengte: waarde in px, em of rem;
- een percentage.

```
p.krap {  
    line-height: 0.9;  
}  
p.normal {  
    line-height: normal;  
}  
p.ruim {  
    line-height: 150%;  
}
```

Regelhoogte

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Provident maiores, veniam veritatis quis! Nisi officiis vel ratione iure eveniet vitae id. Voluptatum neque, fuga optio sint mollitia itaque asperiores a.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Sunt, molestiae quas doloremque tempora. Vero molestiae reprehenderit, doloribus. Ut debitis, ducimus corrupti, delectus preferendis odio et at rem insto labore consequatur.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Autem quasi, quidem rerum ducimus possimus, hic sunt, quo distinctio aperiam fugit quis odio minus porro nemo, dolor quam ea mollitia recusandae.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Id voluptate non at, neque natus eius ad facilis dolor. Ea, reiciens incidentum aliquid doloribus quaerat vero ipsa repudiandae vitae a repellat.

Afbeelding 11.10 Het effect van de eigenschap `line-height`. Van boven naar beneden krap, normal, ruim, normal.



Verkorte notatie

In de verkorte notatie wordt `line-height` gecombineerd met `font-size` als twee waarden met een slash ertussen, bijvoorbeeld zo:

`font: 20px/1.5 Source Sans Pro;`

Zie de paragraaf *De verzameleigenschap font*.

Letterstijl: font-style

De `font-style` bepaalt of tekst normaal, cursief of schuin wordt weergegeven. De waarden zijn `normal` (standaard), `italic` (cursief) of `oblique` (schuingetrokken). Het verschil tussen schuin (`oblique`) en cursief (`italic`) niet altijd even duidelijk te zien. Cursief is een echte variant van het lettertype, terwijl schuin (ook wel: gesleuterd) een enigszins scheef gezette normale letter is. Oblique is ook het ‘cursief’ dat de browser toepast wanneer een echte cursieve variant van een lettertype ontbreekt.

```
p.cursief {  
    font-style: italic;  
}
```

Font-style

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Provident maiores, veniam veritatis quis! Nisi officiis vel ratione iure eveniet vitae id. Voluptatum neque, fuga optio sint mollitia itaque asperiores a.

***L**orem ipsum dolor sit amet, consectetur adipisicing elit. Sunt, molestiae quis doloremque tempora. Vere molestiae reprehenderit, doloribus. Ut debitis, ducimus corrupti, delectus perferendis odio et at rem iusto labore consequatur.*

***L**orem ipsum dolor sit amet, consectetur adipisicing elit. Autem quasi, quidem rerum ducimus possimus, hic sunt, quo distinctio aperiam fugit quis odio minus porro nemo, dolor quam es mollitia recusandae.*

Afbeelding 11.11 De drie letterstijlen van CSS: `normal`, `italic` en `oblique`. Dit is *Times New Roman* en dat systeemlettertype heeft een ‘echte’ italic. Als u goed kijkt ziet u het verschil met oblique.

Lettergewicht: font-weight

Het gewicht (de vetheid) van het lettertype wordt ingesteld met `font-weight`. Tekst kan lichter worden getoond (dunner) of juist zwaarder (vetter). U kunt waarden van 100 tot 900 gebruiken (in stappen van 100), en de waarden `normal` (400, de standaard), `bold` (700), `bolder` (maakt tekst zwaarder dan de voorafgaande tekst) en `lighter` (maakt tekst lichter dan de voorafgaande tekst).

Een voorwaarde voor succes is dat het hier gespecificeerde gewicht ook als lettertypevariant beschikbaar is voor de bezoeker (bijvoorbeeld via de regel `@font-face`). Is dat gewicht er niet, dan wordt de dichtstbijzijnde waarde gekozen. Lukt het helemaal niet om een vette letter weer te geven, controleer dan of de variant `bold` (700) in uw lettertypebibliotheek staat.

Het is overigens niet ongebruikelijk dat een weblettertype alleen beschikbaar is als 300 (light), 400 (normal/regular), 700 (bold) en 900 (black).

```
h2 {  
    font-weight: 900;  
}  
strong {  
    font-weight: 700;  
}
```

Font-weight

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Provident maiores, veniam veritatis quis! Nisi officiis vel ratione iure eveniet vitae id. Voluptatum neque, fuga optio sint mollitia itaque asperiores a.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Sunt, molestiae quas doloremque tempora. Vero molestiae reprehenderit, doloribus. Ut debitis, ducimus corrupti, delectus perferendis odio et at rem iusto labore consequatur.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Autem quasi, quidem rerum ducimus possimus, hic sunt, quo distinctio aperiam fugit quis odio minus porro nemo, dolor quam ea mollitia recusandae.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Autem quasi, quidem rerum ducimus possimus, hic sunt, quo distinctio aperiam fugit quis odio minus porro nemo, dolor quam ea mollitia recusandae.

Afbeelding 11.12 Alle beschikbare gewichten van het lettertype Merriweather: 300, 400, 700 en 900. Hoewel de lettergrootte ook 20px is, is de tekst groter dan in Times New Roman.

De verzameleigenschap font

Met de tot nu toe genoemde eigenschappen kan een complete declaratie voor het lettertype worden gemaakt:

```
p {  
    font-style: italic;  
    font-size: 20px;
```

```
line-height: 1.3;
font-family: 'Times New Roman', Times, serif;
}
```

Het is echter niet nodig om de eigenschappen in aparte stijlregels op te nemen. Ze kunnen ook worden samengevat in de verzameleigenschap font:

```
p {
  font: italic 20px/1.3 'Times New Roman', Times, serif;
}
```

Nogmaals, de volgorde is belangrijk: font-style, font-variant en font-weight moeten komen voor font-size/line-height en font-family. Het is niet verplicht alle eigenschappen te noemen, maar font-size en font-family moeten worden vermeld.

Elke eigenschap die niet wordt opgenomen in de verkorte notatie font – maar dit geldt voor elke verkorte notatie – wordt ingesteld op zijn initiële waarde. Deze waarden vindt u terug in de CSS-specificatie.

Uitrekken of indrukken: font-stretch

Lettertekens kunnen een samengeperste variant (condensed) of ruimere variant (expanded) hebben. In CSS3 kunnen deze varianten worden gebruikt met behulp van de eigenschap font-stretch. De variant van het lettertype die het dichtst bij de aangegeven waarde komt, wordt geladen. Waarden lopen van ultra-condensed via normal tot ultra-expanded (zie www.w3.org/TR/css3-fonts/#propdef-font-stretch).

Om deze lettertypevarianten te kunnen gebruiken, moeten ze voorkomen in de lettertypefamilie en ook daadwerkelijk beschikbaar zijn via de systeemlettertypen of gedownload met behulp van @font-face en dat is bij Google Fonts zelden het geval. Wel zijn vaak van condensed of expanded losse varianten beschikbaar, bijvoorbeeld Roboto en Roboto Condensed. In plaats van font-stretch te gebruiken (het wordt ook niet gesimuleerd door de browser), kan een klasse voor het lettertype worden gemaakt.

```
body {
  font-family: Roboto;
}

.condensed {
  font-family: 'Roboto Condensed';
}
```

Font-stretch

*...lorem ipsum dolor sit amet, consectetur adipisicing elit. Provident maiores, veniam
veritatis quis! Nisi officiis vel ratione iure eveniet vitae id. Voluptatum neque, fuga
optio sint mollitia itaque asperiores a.*

... ipsum dolor sit amet, consectetur adipisicing elit. Provident maiores, veniam veritatis quis! Nisi officiis vel ratione iure eveniet vitae id. Voluptatum neque, fuga optio sint mollitia itaque asperiores a...

Afbeelding 11.13 Geen ingebouwd condensed? Gebruik dan een condensed variant van het lettertype. Hier Roboto Condensed en Roboto.

Kleinkapitaal: font-variant

In CSS3 wordt het aantal mogelijkheden voor font-variant enorm uitgebreid, maar vanwege de beperkte ondersteuning blijven die opties hier onbesproken. Zie www.w3.org/TR/css3-fonts/ voor de – overigens boeiende – toekomstige mogelijkheden.

Eén optie dateert van CSS 2.1 en die is wel bruikbaar: kleinkapitaal, hoofdletters in de grootte van normale tekst.

,kleinkapitaal { font-variant: small-caps; }

Font-variant

**Consectetur adipisicing elit. Provident maiores, veniam
veritatis quis! Nisi officiis vel ratione are eveniet vitae id.**

LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISICING ELIT. PROVIDENT MAJORES, VENIAM VERITATIS QUIS!? NISI OFFICIS VEL RATIONE IURE EVENIET VITIAE ID.

Afbeelding 11.14 Kleinkapitaal is zoals u ziet niet geschikt voor gewone tekst, maar tussenkoppen en afkortingen zijn goede toepassingen. En u kunt beschaafd schreeuwen op sociale media.



Meer font-eigenschappen

De meestgebruikte en best ondersteunde eigenschappen voor font zijn besproken, maar er zijn er nog meer. Kijk op developer.mozilla.org/en-US/docs/Web/CSS/font.

Eigenschappen voor tekst

In de voorgaande paragraaf zijn eigenschappen behandeld die rechtstreeks samenhangen met het lettertype. Daarnaast is het mogelijk tekst (tekens, woorden en alinea's) op te maken. Denk hierbij aan witruimte, uitlijning en ingesprongen tekst, maar ook aan tekstversiering en slagschaduw.

- `text-align` (uitlijnen)
- `text-align-last` (uitlijning laatste regel)
- `text-decoration` (versiering)
- `text-indent` (inspringen)
- `text-orientation` (horizontale of verticale tekst)
- `text-overflow` (overlopende tekst)
- `text-shadow` (schaduw)
- `text-transform` (hoofdletters, kleine letters)
- `white-space` (witruimte)
- `word-spacing` en `letter-spacing` (witruimte)
- `word-break` (afbreken)
- `word-wrap` (afbreken)

Uitlijnen: `text-align` en `text-align-last`

Tekst kan worden uitgelijnd tegen de linker- of rechterkant, worden gecentreerd of worden uitgevuld. De eigenschap daarvoor is `text-align`. Waarden zijn `start` (standaard), `end`, `left`, `right`, `center`, `justify` (uitvullen) en `match-parent` (afbeelding 11.15).

Met deze eigenschap kunt u tekst in afzonderlijke alinea's uitlijnen, maar ook alle alinea's in een container.

Voor het uitlijnen van de laatste regel van een alinea is er de eigenschap `text-align-last`. Waarden zijn `auto` (standaard), `start`, `end`, `left`, `right`, `center` en `justify`.

Lijneffecten op tekst

Voor lijneffecten bij tekst bestaat sinds CSS2.1 de eigenschap `text-decoration` met de waarden voor lijnen onder, door en boven de tekst. In CSS3 (de module *Text Decoration level 3*) is `text-decoration` een verkorte eigenschap geworden voor `text-decoration-line`, `text-decoration-color` en `text-decoration-style`. Het aantal mogelijkheden is daarmee aanmerkelijk groter.

Uitlijning

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Provident maiores, veniam veritatis quis! Nisi officiis vel ratione iure eveniet vitae id. Voluptatum neque, fuga optio sint mollitia itaque asperiores a.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Provident maiores, veniam veritatis quis! Nisi officiis vel ratione iure eveniet vitae id. Voluptatum neque, fuga optio sint mollitia itaque asperiores a.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Provident maiores, veniam veritatis quis! Nisi officiis vel ratione iure eveniet vitae id. Voluptatum neque, fuga optio sint mollitia itaque asperiores a.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Provident maiores, veniam veritatis quis! Nisi officiis vel ratione iure eveniet vitae id. Voluptatum neque, fuga optio sint mollitia itaque asperiores a.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Provident maiores, veniam veritatis quis! Nisi officiis vel ratione iure eveniet vitae id. Voluptatum neque, fuga optio sint mollitia itaque asperiores a.

Afbeelding 11.15 Voorbeelden van uitlijning: start, center, end, justify en onderaan justify met align-last-line: justify.

Text-decoration

Lorem ipsum dolor sit amet. text-decoration: underline wavy red;

Lorem ipsum dolor sit amet. text-decoration: overline dotted green;

Lorem ipsum dolor sit amet. text-decoration: line-through solid red;

Lorem ipsum dolor sit amet. text-decoration: underline double purple;

Lorem ipsum dolor sit amet. text-decoration: underline overline dashed blue;

Afbeelding 11.16 Varianten van text-decoration. Ook dubbele waarden zijn mogelijk (zie onderaan).

Onderstrepen of doorhalen (met de korte eigenschap `text-decoration`) werkt vanwege de compatibiliteit met CSS 2.1 altijd; de ardere effecten leiden soms nog aan beperkte ondersteuning.



Hyperlinks zonder onderstrekking

De onderstrekking van hyperlinks komt ook gewoon uit de eigenschap `text-decoration`. Een declaratie voor een link zonder onderstrekking is `a { text-decoration: none; }`. Als u de onderstrekking weghaalt, moet u de hypelink wel op een andere manier onderscheiden van gewone tekst.

Hieronder staan de afzonderlijke eigenschappen met hun waarden. Deze volgorde is ook de volgorde in de verkorte eigenschap.

`text-decoration-line`

- `none` (standaard)
- `overline`
- `underline`
- `strike-through`

`text-decoration-style`

- `solid`
- `double`
- `dotted`
- `dashed`
- `wavy`

`text-decoration-color`

- Elke geldige kleurwaarde (zie hoofdstuk 12).

Inspringen

De eerste opgemaakte regel van een tekstblok kan inspringen met de eigenschap `text-indent`. De standaardwaarde is 0 en mogelijke waarden zijn een lengte in px, em of rem, of een percentage:

De voorbeeldcode toont een alinea met standaard ingesprongen tekst en hangend ingesprongen tekst, wat wordt bereikt door een marge en negatieve inspring toe te passen.

```
p { /* eerste regel springt 1em in */
  text-indent: 1em;
}
```

```
p { /* hangend inspringen */
margin-left: 1em;
text-indent: -1em;
}
```

Text-indent

 Lorem ipsum dolor sit amet, consectetur adipisicing elit. Provident maiores, veniam veritatis quis! Nisi officiis vel ratione iure eveniet vitae id. Voluptatum neque, fuga optio sint mollitia itaque asperiores a.

 Lorem ipsum dolor sit amet, consectetur adipisicing elit. Provident maiores, veniam veritatis quis! Nisi officiis vel ratione iure eveniet vitae id. Voluptatum neque, fuga optio sint mollitia itaque asperiores a.

Afbeelding 11.17 Ingesprongen tekst, normaal en andersom.



Nieuwe waarden, maar geen ondersteuning

Nieuwe sleutelwoorden zijn `hanging` en `each-line`, waarmee respectievelijk het effect wordt omgedraaid (hangend inspringen) en elke eerste regel na een geforceerd regeleinde in een container inspringt. Deze zijn nog niet bruikbaar.

Rechtopstaande tekst: text-orientation

De eigenschap `text-orientation` is bedoeld voor talen die niet horizontaal geschreven worden. Desondanks is het bruikbaar in Latijns schrift, mits de eigenschap `writing-mode` wordt ingesteld op `vertical-rl` of `vertical-lr` (verticaal links-rechts of rechts-links – onze Latijnse standaardinstelling is `horizontal-tb`, wat staat voor top-bottom). Een mogelijk toepassing is een

L
o
r
e
m

i
p
s
u
m

F
e
e
d
b
a
c
k

Afbeelding 11.18 Rechtopstaande tekst met writing-mode en text-orientation.

rechtopstaande tabelkop of een feedbackknop aan de rand van het venster. Zo'n tekst maakt u bijvoorbeeld zo:

```
p {  
writing-mode: vertical-rl;  
text-orientation: upright;  
}
```

Overlopende tekst: text-overflow

Er is een overloopeigenschap speciaal voor tekst: `text-overflow`. Met deze eigenschap kunt u bepalen welk visueel signaal de gebruiker krijgt dat er meer tekst is dan hij ziet. In theorie zijn er verschillende mogelijkheden, maar in de praktijk werkt er één: een markering met drie puntjes...

Het is een beetje een vreemde eigenschap, want het werkt alleen in de regel. In de vorige paragraaf zag u tekst die onderaan uit de container loopt, maar dat is geen `text-overflow`. Er kan alleen `text-overflow` zijn als de container de eigenschap `overflow: hidden` heeft en als `white-space: nowrap` is ingesteld (verhindert dat tekst afbreekt op spaties, zie hoofdstuk 9). Pas dan kunt u `text-overflow` instellen met:

- `clip` (standaard) de tekst wordt afgekapt en meer niet.
- `ellipsis` aan het eind van de zichtbare tekst worden drie puntjes getoond ... (een wegatingsteken in het Nederlands).

`... Lorem ipsum dolor sit amet, consectetur adip...`

Afbeelding 11.19 *Er is meer tekst, maar die is afgekapt. Dat is te zien aan de drie puntjes.*

Tekst met schaduw

Het is mogelijk tekst te voorzien van een schaduw met `text-shadow`. De standaardwaarde is `none`. Voor een schaduw definieert u minimaal één en maximaal drie waarden:

- horizontale verschuiving
- verticale verschuiving
- vervaging
- optioneel een kleur

Positieve waarden voor horizontaal en verticaal verplaatsen de schaduw respectievelijk naar rechts en omlaag; negatieve waarden naar links en omhoog. Een schaduw naar rechts en omlaag is bijvoorbeeld:

```
text-shadow: 4px 4px 8px rgba(0,0,0,0.2);
```



Kleurwaarden

Kleuren kunnen in CSS op diverse manieren worden ingesteld. Dit wordt uitgelegd in hoofdstuk 12.

Leuke effecten zijn ook mogelijk door de verschuiving op 0 te zetten, bij een tekstkleur die hetzelfde is als de achtergrondkleur en een zwarte schaduw. Daarmee wordt een outline-effect gecreëerd. Het is ook mogelijk om verschillende schaduwen op een element te zetten. Plaats een komma na elk groep schaduwwaarden en voeg meer waarden toe.

Tekstschaduw

```
text-shadow: .2em .2em .5em black;
```

Tekstschaduw

```
text-shadow: 0 1px 1px #fff;
```

Tekstschaduw

```
text-shadow: 0 0 2px black, 0 0 2px black, 0  
0 2px black;
```

Tekstschaduw

```
text-shadow: 0 0 .1em, 0 0 .3em;
```

Afbeelding 11.20 Door te variëren met waarden en kleuren zijn er eindeloos veel mogelijkheden met tekstschaduw.

Tekst omzetten: text-transform

Ongeacht of tekst in hoofdletters of kleine letters in de HTML staat, kan deze met de eigenschap `text-transform` worden omgezet in wat u wilt. Waarden zijn:

- `capitalize` elke eerste letter van een woord wordt een hoofdletter, de rest blijft hetzelfde;
- `uppercase` alle tekens worden omgezet in hoofdletters;
- `lowercase` alle tekens worden omgezet in kleine letters;
- `none` de standaardwaarde, er gebeurt niets.

Er wordt nog gesleuteld aan verfijning voor typische verschijnselen in diverse talen, zoals de hoofdletter `IJ` in het Nederlands of de hoofdletter `ringels` in het Duits, maar niet alle browsers kunnen dat. Dan nog werkt het alleen als het document of het element een corresponderende taalinstelling heeft.



Tekst transformeren

Misschien had u hogere verwachtingen van `text-transform`, zoals het kunnen draaien en scheeftrekken van tekst. Ook dat is mogelijk, maar het valt onder een andere noemer en er kan ook meer mee dan alleen tekst transformeren. De eigenschap (eigenlijk een functie) heet `transform()`.

Text-transform

`lorem ipsum dolor sit amet, consectetur adipisicing elit.`

`LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISICING ELIT`

`IJsbrand Eet IJsjes Met Islandse Meisjes.`

Afbeelding 11.21 In Firefox kan `text-transform` een hoofdletter `IJ` maken. Chrome doet nog niet mee.

Witruimte behouden: white-space

Witruimte in HTML-teks wordt standaard niet behouden, maar u hebt ook geleerd dat met het element `<pre>` het wit (spaties en tabs, returns) in vooraf opgemaakte tekst wel kan worden behouden. Er is ook een CSS-eigenschap waarmee dit kan. De eigenschap `white-space` biedt daarbij meer mogelijkheden dan het HTML-element. Mogelijke waarden zijn:

- `normal` wit wordt samengevoegd tot één spatie, regeleinden vervallen en de tekstomloop is normaal;

- `pre` al het extra wit en extra regeleinden blijven behouden, geen tekstomloop;
- `nowrap` wit wordt samengevoegd, regeleinden vervallen, geen tekstomloop;
- `pre-wrap` wit en regeleinden blijven behouden, normale tekstromloop;
- `pre-line` wit wordt samengevoegd, regeleinden blijven behouden, tekstomloop normaal.

Bedenk dat bij het ontbreken van tekstromloop de tekst doorloopt buiten de container als die te smal is (de container is het blok dat de alineatekst omsluit). Of die tekst dan zichtbaar blijft, hangt af van de eigenschap `overflow` van de container (standaard zichtbaar, zie ook hoofdstuk 9).



Tabsprong: tab-size

Elke tab in HTML is in opgemaakte tekst met een instelling voor het behoud van witruimte standaard een sprong ter grootte van 8 spaties in het ingestelde lettertype. U kunt deze waarde aanpassen door de eigenschap `tab-size` in te stellen op een ander getal. Controleer ondersteuning op caniuse.com.

Witruimte

 Lorem ipsum dolor sit amet, consectetur
 adipisicing elit. Provident maiores, veniam
 veritatis quis!

 Lorem ipsum dolor sit amet, consectetur ...

 Lorem ipsum dolor sit amet,
 consectetur adipisicing elit. provide...
 veniam veritatis quis! Nisi offic...
 iure eveniet vitae id. Volu...

Afbeelding 11.22 Wat gebeurt er met spaties en tabs? Boven normaal, midden nowrap met ellips en onder pre met tabs in de brontekst en ellipsen waar de tekst is afgekapt.

Ruimte tussen woorden en letters

Met behulp van `word-spacing` en `letter-spacing` wordt de ruimte tussen woorden en letters ingesteld. Beide eigenschappen kunnen dezelfde waarden hebben: `normal` (standaard) of een lengte (`px`, `em` of `rem`). De opgegeven waarde wordt toegevoegd aan de eigen witruimte van het lettertype. Negatieve waarden verkleinen de witruimte.

Spatiēring

...lorem ipsum dolor sit amet, consectetur adipisicing elit. Provident maiores, veniam veritatis quis! Nisi officiis vel ratione iure eveniet vitae id. Voluptatum neque, fuga optio sint mollitia itaque superiores a.

Consectetur adipisicing elit. Provident maiores, veniam veritatis quis! Nisi officiis vel ratione irre eveniet vitae id. Voluptatum neque, fuga optio sint mollitia itaque asperiores a.

...et asperiores.

Afbeelding 11.23 Meer en minder luchtige tekst met word-spacing en letter-spacing. Van boven naar beneden meer word-spacing, meer letter-spacing en van beide een pixel minder.

Afbreken: word-break en overflow-wrap

In het deel over HTML vindt u dat met het element <wbr> afbreekpunten in lange woorden worden ingesteld. Dat moet met de hand gebeuren en kan lastig zijn. Automatisch afbreken kan met word-break en overflow-xrap . (overflow-wrap is de nieuwe naam van word-wrap. De oude naam blijft geldig.)

Geef word-break de instelling break-all en lange woorden worden afgebroken als ze niet meer aan het eind van de regel passen. Bij overflow-wrap en de instelling break-word wordt alleen afgebroken als het woord echt niet in de container past, ook niet op een nieuwe regel (afbeelding 11.23).

Afbreekstreepjes met hyphens

De eigenschap `hyphens` kan afbreekstreepjes toevoegen, maar de browserondersteuning is nog niet volledig. Als de eigenschap wel werkt is het altijd nog beter dan afbreken zonder streepjes, dus dat kan een reden zijn om hyphens te gebruiken en dan maar te zien wat ervan komt.

De beste kan op succes geeft hyphens: auto, waarbij de browser het zelf uitzoekt. Het is altijd verstandig om in de head van de pagina aan het element <html> het attribuut lang="nl" toe te voegen, maar voor zinnig afbreken is het een vereiste.

Langewoordenscrabble

In onze dolgedraaide vierentwintiguruseconomie is het van groot belang dat wij de begrotingsdiscipline bewaken.

In onze dolgedraaide vierentwintiguruseconomie is het van groot belang dat wij de begrotingsdiscipline bewaken.

In onze dolgedraaide vierentwintiguruseconomie is het van groot belang dat wij de begrotingsdiscipline bewaken.

In onze dolgedraaide vierentwintiguruseconomie is het van groot belang dat wij de begrotingsdiscipline bewaken.

Afbeelding 11.24 De stippellijn is de outline van de alinea.

1. word-break: initial en overflow-wrap: initial
2. alleen word-break: break-all
3. alleen overflow-wrap: break-word
4. overflow-wrap: break-word en word-break: break-all.

Netjes afbreken volgens de ingestelde taal zit er nog niet in, afbrekstreeppjes cok niet.

Opmaak van lijsten

Voor de opmaak van lijsten – en dan denken we ook aan de navigatie – zijn eigenschappen beschikbaar om het opsommingsteken aan te passen (of te laten vervallen), een eigen afbeelding te gebruiken en om de positie van het teken te veranderen. De afzonderlijke eigenschappen zijn:

- list-style-type
- list-style-image
- list-style-position

Voor het instellen van een of meer eigenschappen ineens kan ook de korte notatie list-style worden gebruikt.

Opsommingstekens

Voor een lijst kan een opsommingsteken uit diverse externe bronnen worden gekozen, maar er zijn in de browser meer mogelijkheden dan de bekende bullet. De eigenschap `list-style-type` bevat een flinke reeks mogelijkheden die van toepassing zijn op `` of `` (nummering op een niet-geordende lijst is niet logisch, maar het is wel mogelijk). De belangrijkste zijn:

- opsommingstekens disc (standaard), circle en square;
- cijfers decimal, decimal-leading-zero
- letters lower-latin, upper-latin, lower-alpha, upper-alpha
- Romeinse cijfers lower-roman, upper-roman

In de modules CSS Lists level 3 (www.w3.org/TR/css-lists-3/) en CSS Counter Styles level 3 (www.w3.org/TR/css-counter-styles-3/) worden meer mogelijkheden genoemd, maar die zijn nog niet bruikbaar en hebben veelal betrekking op niet-Latijnse talen.

Een genummerde lijst met voorloopnul codeert u als:

```
ol {
    list-style-type: decimal-leading-zero;
}
```



Afbeelding 11.25 Diverse mogelijkheden met opsommingstekens: genummerd met voorloopnul, een eigen afbeelding en het verschil tussen outside (standaard) en inside.

Een eigen afbeelding in plaats van een van de ingebouwde mogelijkheden kan ook. Daarvoor gebruikt u de eigenschap `list-style-image`:

```
ul {  
    list-style-image: url('../images/leaf.svg');  
}
```

Standaard staat het opsommingsteken buiten het tekstblok, maar het kan er ook in gezet worden met de eigenschap `list-style-position`. Waarden zijn `outside` (standaard) en `inside`. Voor items in een lijst staat standaard zo'n 40 pixels padding. In die padding wordt het teken `outside` geplaatst. Is de plaatsing `inside` dan staat de bullet in het tekstblok. Dan wordt de lijst breder, want de padding blijft hetzelfde, tenzij u die aanpast. De padding zit op de ``.

Lijst als navigatie

Er is eindeloos veel opmaak te bedenken voor navigatie die bestaat uit lijst-items, maar in de basis draait het maar om een paar dingen:

- het opsommingsteken moet weg;
- de 40 pixels padding links moet weg;
- de marge boven en onder moet weg.

Verder zijn er in dit voorbeeld twee aanvullende eisen:

- op mobiel staan de items onder elkaar;
- op grotere schermen staan de items naast elkaar.

Hoe het er met opmaak ook uitziet, de HTML blijft hetzelfde. Neem de volgende code over in een nieuw HTML-bestand. Maak ook een map met de naam `css`, want daar moet de stylesheet in.

- De uitgewerkte code is te vinden in `11_01.html`.

```
<!DOCTYPE html>  
<html lang="nl">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <title>11_01</title>  
    <link href="https://fonts.googleapis.com/css?family=Roboto:400,400i,500" rel="stylesheet">  
    <link rel="stylesheet" href=".//css/11_01.css">
```

```
</head>
<body>
<nav>
  <ul>
    <li><a href="">Home</a></li>
    <li><a href="">Nieuws</a></li>
    <li><a href="">Producten</a></li>
    <li><a href="">Over ons</a></li>
    <li><a href="">Contact</a></li>
  </ul>
</nav>
</body>
</html>
```

Stap 1: basisopmaak

In de `<head>` van het HTML-bestand laden we het lettertype Roboto van Google Fonts en wordt aangegeven dat de opmaak is te vinden in de stylesheet `11_01.css`.

Maak de stylesheet in de map `css`. Het lettertype wordt ingesteld op `body`, zodat het voor alle elementen geldt.

```
body {
  font: 20px/1.5 Roboto, sans-serif;
}
```

Open het bestand in de browser en open de ontwikkelaarshulpmiddelen met `Ctrl+Shift+I`. Klik in Chrome op de tab `Elements` en klik in de HTML-weergave op het element ``. Nu is bij `Styles` te zien welke CSS-eigenschappen `` heeft: `list-style-type`, `margin` en `padding`. Deze komen uit de *user agent stylesheet*, de ingebouwd stylesheet van de browser. Van al deze eigenschappen willen we af, want die zitten maar in de weg bij een menu:

```
ul {
  list-style: none;
  margin: 0;
  padding: 0;
  width: max-content;
}
```

De lijst is nu schoon, maar de hyperlinks zijn nog onderstreept en dat is in het menu niet de bedoeling. Het is ook niet de bedoeling dat de kleur van de (bezochte) hyperlinks verandert. Met de pseudo-klasse `:any-link` worden tegelijk `:link` en `:visited` beïnvloed:

```
a:any-link,  
a:active {  
    text-decoration: none;  
    color: inherit;  
}
```

Het resultaat van stap 1 is een schone lijst van menu-items.

Stap 2: het menu responsive maken

Het menu is een lijst, dus de items staan onder elkaar en op mobiel is dat goed. Maar als het venster breder wordt, moeten de items naast elkaar staan. Nu is een iPad in portretstand 768px breed en die waarde gebruiken we in een media query. De items worden naast elkaar gezet door van de lijst een flexcontainer te maken.

```
@media screen and (min-width: 768px) {  
    ul {  
        display: flex;  
    }  
}
```

Nu blijken de items erg dicht op elkaar te staan. Met padding op de items wordt dat opgelost.

```
@media screen and (min-width: 768px) {  
    ul {  
        display: flex;  
    }  
    li {  
        padding: 10px;  
    }  
}
```

Stap 3: het menu tonen en verbergen

Het is niet handig als op mobiel het menu altijd in beeld staat; dat kost maar ruimte. Beter is een menuknop waarmee de items worden getoond en verborgen. Er zijn duizend en een manieren om zoets te maken, ook met JavaScript, maar in dit voorbeeld doen we alles met HTML en CSS.

De techniek draait om een onzichtbaar selectievakje, de eigenschap checked van dat vakje en een <label>. Als het selectievakje is ingeschakeld, wordt het menu getoond; als het vakje uitstaat is het menu verborgen.

We breiden eerst de HTML-code uit:

```
<nav>
  <label for="menu-toggle">menu</label>
  <input type="checkbox" id="menu-toggle">
  <ul>
    <li><a href="">Home</a></li>
    <li><a href="">Nieuws</a></li>
    <li><a href="">Producten</a></li>
    <li><a href="">Over ons</a></li>
    <li><a href="">Contact</a></li>
  </ul>
</nav>
```

Het selectievakje wordt verborgen:

```
#menu-toggle {
  display: none;
}
```

Het vakje is nu verdwenen, maar het label blijft zichtbaar en omdat het label via de ID is gekoppeld aan het vakje, kan dat nog steeds worden in- en uitgeschakeld.

Bij stap 1 is de CSS voor `` ingesteld. Omdat het menu nu eerst moet worden verborgen, breiden we de declaratie uit:

```
ul {
  display: none;
  list-style: none;
  margin: 0;
  padding: 0;
}
```

Een selectievakje staat standaard uit nadat de pagina is geladen. Het menu wordt zichtbaar gemaakt nadat het is vakje ingeschakeld door op het label te klikken; dan krijgt het de status `checked`.

```
}
```

```
input:checked ~ ul {
  display: initial;
}
```

Hier staat: selecteer een element dat een sibling is van een <input> met de status checked en maak dat element zichtbaar door het de initiële waarde te geven. We willen immers dat het menu een lijst is. Bekijk de pagina in de browser en klik op het woord menu. De menulijst wordt afwisselend getoond en verborgen.

Als het venster breder is, moet het label onzichtbaar zijn en het menu juist zichtbaar. Daarvoor wordt de media query aangepast. Let op: ook als het selectievakje is ingeschakeld moet het menu een balk zijn, geen lijst. Daarvoor is het niet voldoende om alleen display: flex te geven. Dat komt doordat de selector input:checked ~ ul een hogere specificiteit heeft dan alleen ul (zie hoofdstuk 8, *Specificiteit berekenen*). Om display: initial te overschrijven moet ook de checked-selector in de media query worden opgenomen.

Specificiteit van de selectoren

ul	een element = 0, 0, 0, 1
input:checked ~ ul	twee elementen en een pseudoklasse = 0, 0, 1, 2

```
@media screen and (min-width: 768px) {
  input:checked ~ ul,
  ul {
    display: flex;
  }
  li {
    padding: 10px;
  }
  [for="menu-toggle"] {
    display: none;
  }
}
```

Met de attribuutselector wordt het label onzichtbaar gemaakt. Varieer de breedte van het browservenster en u ziet dat:

- op mobiel het menu verborgen is, heeft display: none;
- op mobiel na een tik op menu de menulijst wordt getoond, heeft display: initial;
- op een breder scherm het label wordt verborgen en het menu als balk wordt getoond, heeft display: flex.

Stap 4: het menu vloeiend openen en sluiten

In modern webdesign wordt veel gebruikgemaakt van animatie en vloeiende bewegingen. Ons menu werkt, maar wel een beetje lomp. Dat kan met een

kleine aanpassing al veel beter. Eerst wordt het menu buiten het venster geplaatst. Bij het openen schuift het met een kleine vertraging in beeld. Daardoor zijn `display: none` en `display: initial` niet meer nodig. Eerst wordt `` buiten beeld gezet.

```
ul {
    /*display: none;*/
    list-style: none;
    margin: 0;
    padding: 0;
    width: max-content;
    transform: translateX(-100px);
    transition: transform 250ms ease-out;
}
```

Dan wordt `` bij tikken op het menulabel in beeld geschoven:

```
input:checked ~ ul {
    /*display: initial;*/
    transform: translateX(0);
    transition: transform 200ms ease-in;
}
```

De menulijst wordt met `translateX 100px` naar links geschoven en staat daarmee buiten beeld. Als het menu wordt geactiveerd schuift het naar de oorspronkelijke plek, dan is de verschuiving 0. Als het wordt gesloten schuift het ook weer vloeiend buiten beeld. `transition` en `transform` worden besproken in hoofdstuk 13.

Tot slot: om te voorkomen dat op een breder scherm het menu 100px naar links staat, wordt de media query aangepast:

```
@media screen and (min-width: 768px) {
    input:checked ~ ul,
    ul {
        display: flex;
        transform: translateX(0);
    }
    li {
        padding: 10px;
    }
    [for="menu-toggle"] {
        display: none;
    }
}
```

Stap 5: doe er wat leuks mee

Het menu is nog kaal, maar daar kunt u zelf wat aan doen. Wat suggesties:

- voeg tekst- en achtergrondkleur toe;
- vervang de tekst menu van het label door een pictogram (zie fontawesome.com, ook voor instructies);
- Voeg aan de menuopties passende pictogrammen toe, zoals in de afbeelding bijvoorbeeld.

Samenvatting

In dit hoofdstuk zijn eigenschappen besproken die te maken hebben met het lettertype (font) en specifieke tekstininstellingen (text). Ook de opmaak van lijsten is behandeld.

- Goede lettertypen kunnen gratis en veilig van internet worden gebruikt.
- Het lettertype van een element wordt ingesteld met font-family, de grootte met font-size. Andere veelgebruikte eigenschappen zijn font-weight (vet) en font-style (cursief).
- Bij teksteigenschappen gaat het bijvoorbeeld om uitlijnen (text-align) en inspringen (text-indent).
- Witruimte kan worden behouden met de eigenschap white-space (denk aan het element `<pre>`).
- Ruimte tussen letters en woorden wordt ingesteld met word-spacing en letter-spacing.
- De eigenschap text-overflow bepaalt wat er met overlopende tekst gebeurt. Een elegante oplossing voor afgekaptte tekst is het beletselteken (...).
- Onderstreping van bijvoorbeeld hyperlinks zit in de eigenschap text-decoration. Met de waarde none is de onderstreping weg.
- Schaduw om tekst (text-shadow) kan onder meer worden gebruikt om in plaats van de letter zelf een omtrek te tonen die alleen bestaat uit schaduw.
- Van een lijst kan het opsommingsteken volledig naar eigen inzicht worden aangepast.
- Voor de navigatie wordt een lijst ontdaan van opsommingstekens en extra witruimte, en naar keus horizontaal of verticaal weergegeven.

Het volgende hoofdstuk draait volledig om kleur en afbeeldingen: voorgrondkleur, achtergrondkleur, transparantie, kleurverlopen (*gradient*), achtergrondafbeeldingen en versiering van randen.

Stap 5: doe er wat leuks mee

Het menu is nog kaal, maar daar kunt u zelf wat aan doen. Wat suggesties:

- voeg tekst- en achtergrondkleur toe;
- vervang de tekst menu van het label door een pictogram (zie fontawesome.com, ook voor instructies);
- Voeg aan de menuopties passende pictogrammen toe, zoals in de afbeelding bijvoorbeeld.

Samenvatting

In dit hoofdstuk zijn eigenschappen besproken die te maken hebben met het lettertype (font) en specifieke tekstininstellingen (text). Ook de opmaak van lijsten is behandeld.

- Goede lettertypen kunnen gratis en veilig van internet worden gebruikt.
- Het lettertype van een element wordt ingesteld met font-family, de grootte met font-size. Andere veelgebruikte eigenschappen zijn font-weight (vet) en font-style (cursief).
- Bij teksteigenschappen gaat het bijvoorbeeld om uitlijnen (text-align) en inspringen (text-indent).
- Witruimte kan worden behouden met de eigenschap white-space (denk aan het element `<pre>`).
- Ruimte tussen letters en woorden wordt ingesteld met word-spacing en letter-spacing.
- De eigenschap text-overflow bepaalt wat er met overlopende tekst gebeurt. Een elegante oplossing voor afgekaptte tekst is het beletselteken (...).
- Onderstreping van bijvoorbeeld hyperlinks zit in de eigenschap text-decoration. Met de waarde none is de onderstreping weg.
- Schaduw om tekst (text-shadow) kan onder meer worden gebruikt om in plaats van de letter zelf een omtrek te tonen die alleen bestaat uit schaduw.
- Van een lijst kan het opsommingsteken volledig naar eigen inzicht worden aangepast.
- Voor de navigatie wordt een lijst ontdaan van opsommingstekens en extra witruimte, en naar keus horizontaal of verticaal weergegeven.

Het volgende hoofdstuk draait volledig om kleur en afbeeldingen: voorgrondkleur, achtergrondkleur, transparantie, kleurverlopen (*gradient*), achtergrondafbeeldingen en versiering van randen.

Kleur, randen en achtergronden

Op deze boekpagina's zal het altijd grijs blijven, maar op uw webpagina's kunt u alle kanten op met kleur, van beschaafd getint tot woest psychedelisch. De mogelijkheden voor het werken met kleur, maar ook die voor achtergronden en randen, zijn de laatste jaren flink uitgebreid. Kleurverlopen en randen met afgeronde hoeken behoren nu tot de standaardwerkset. Boxen kunnen een schaduwrand krijgen en het is mogelijk om afbeeldingen uit te snijden en vormen te maken. Tot slot kijken we naar filters voor beeldbewerking in de browser en eigenschappen voor het mengen van (achtergrond)kleuren.

U leert in dit hoofdstuk:

Welke kleursystemen en kleurwaarden CSS ondersteunt.

De achtergronden van elementen vormgeven met kleur en afbeeldingen.

Kleurverlopen gebruiken: lineair, radiaal en repeterend.

Randen opmaken met afbeeldingen.

Schaduweffecten toepassen op blokken.

Afbeeldinger uitsnijden en vormen maken met clip-path.

Beeldfilters toepassen en kleuren mengen met blend-mode.

Inleiding

Dit hoofdstuk gaat meer in op het uiterlijk vertoon van CSS: kleur en afbeeldingen. Het onderdeel kleur is in de loop der jaren flink uitgebreid en naast de traditionele systeemkleuren en RGB-kleuren worden nu ook transparantie en het HSL-kleursysteem ondersteund.

Dat elementen een rand hebben weet u sinds hoofdstuk 9. Dat die randen kunnen worden versierd met slagschaduw, afbeeldingen en ronde hoeken leest u hier.

Ook de achtergronden van elementen komen aan bod. Elk elementen heeft standaard een transparante achtergrond, maar dat kan elke kleur zijn en ook een of meer afbeeldingen. Sterker nog, een achtergrondafbeelding hoeft geen bitmap te zijn, een kleurverloop kan ook. Hoe kleurverlopen worden gemaakt wordt dan natuurlijk ook uitgelegd.

Daarna komen we terug op de randen, maar nu met de uitgebreide mogelijkheden voor het opmaken ervan: afgeronde hoeken, randen met afbeeldingen en schaduw.

Kleurwaarden

Voor het instellen van kleuren worden de CSS-kleurwaarden gebruikt. Dat waren ooit alleen 'webveilige' kleurnamen van de zestien kleuren die op alle destijds eenvoudige beeldschermen te zien waren. In de jaren erna zijn de mogelijkheden uitgebreid met RGB-kleuren, en met CSS3 kwam het HSL-kleursysteem erbij, plus transparantie voor zowel RGB als HSL. Intussen is ook het systeem van kleurnamen nog uitgebreid tot ongeveer 150 benoemde kleuren.

Omdat er verschillende kleursystemen zijn, kan een kleureigenschap ook op verschillende manieren zijn kleurwaarde krijgen. Het volgende rijtje bestaat uit verschillende declaraties voor een niet-transparante rode kleur:

color: red	de kleurnaam
color: #ff0000	RGB met hexadecimale notatie: rrggbb
color: #f00	RGB in verkorte hexadecimale notatie: rgb
color: rgb(255, 0, 0)	RGB met decimale notatie
color: rgb(100%, 0, 0)	RGB met waarde in procenten
color: rgba(255, 0, 0, 1)	RGB met transparantie
color: hsl(0, 100%, 50%)	het HSL-systeem
color: hsla(0, 100%, 50%, 1)	het HSL-systeem met transparantie

Voldoende keus... maar wat is de juiste, de beste of de handigste?

Kleurnamen kunnen handig zijn als u aan het ontwerpen bent, even snel een rand of een outline instellen met `outline` of `border: 1px solid black` vraagt eigenlijk geen denkwerk. Met een goed overzicht van de kleurnamen zijn ze natuurlijk ook prima en snel te gebruiken in uw ontwerpen. Handig is developer.mozilla.org/en-US/docs/Web/CSS/color_value.

Bent u niet bekend bent met het hexadecimale talstelsel, dan kunt u dat gelijk vergeten. Het is niet de moeite om dat te leren alleen voor CSS-kleuren, want er zijn handiger methoden. Bent u er toch in geïnteresseerd, lees dan de opmerking. Wilt u zich echt verdiepen in hexadecimale kleur, lees dan dit artikel: www.smashingmagazine.com/2012/10/04/the-code-side-of-color/.



Hexadecimaal

Het hexadecimale stelsel is een zestientallig stelsel, dat loopt van 0 tot F. Daarbij is A gelijk aan 10 decimaal, B aan 11 enzovoort. Een getal wordt opgebouwd in paren, waarbij het rechtergetal in het paar loopt van 0 tot 15 en het linkergetal van 0 tot n maal 16. Enkele voorbeelden: $01 = 1 (1 \times 1)$, $10 = 16 (1 \times 16 + 0)$, $1F = 31 (1 \times 16 + 15 \times 1)$, $FF = 255 (15 \times 16 + 15 \times 1)$. Het maakt niet uit of u hoofdletters of kleine letters gebruikt. Een hexadecimaal getal wordt in CSS voorafgegaan door een #, bijvoorbeeld: #8c448f. Een hexadecimale kleur waarvan elk paar uit dezelfde waarden bestaat, mag worden afgekort. Zo is #ff0000 hetzelfde als #f00.

RGB en RGBA

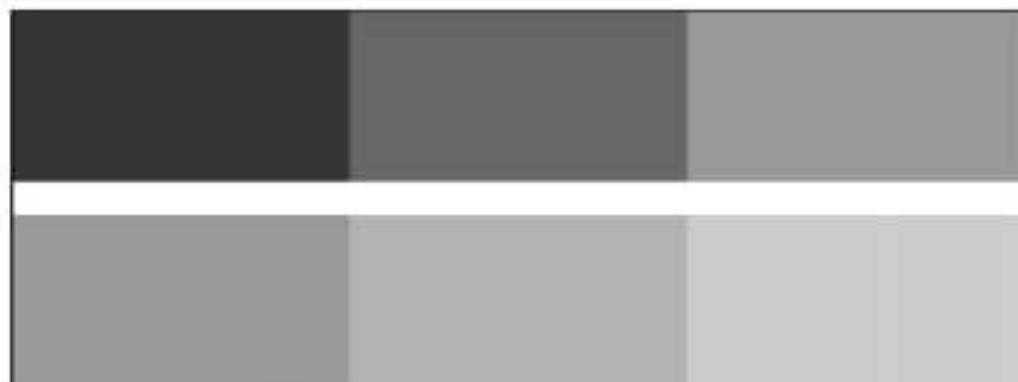
Of u kiest voor RGB of HSL hangt vooral af van wat u gewend bent. Wie al zijn hele leven met RGB-kleuren werkt, zal vertrouwd zijn met het mengen van rood, groen en blauw door de waarde voor elke kleur te variëren tussen 0 en 255.

- `rgb(0, 0, 0)` is zwart;
- `rgb(255, 0, 0)` is rood;
- `rgb(0, 255, 0)` is groen;
- `rgb(0, 0, 255)` is blauw;
- `rgb(255, 255, 255)` is wit;
- `rgb(10, 160, 255)` is een blauwtint.

Hoe hoger de waarde, hoe lichter de tint. Als alle waarden hetzelfde zijn ontstaat een grijstint. Hogere gelijke waarden geven een lichtere grijstint.

De `a` in `rgba` voegt transparantie toe en komt van alfa – denk aan het alfakanalen in Photoshop. De waarde ligt tussen 0 en 1 en gaat van volledig doorzichtig tot de volledig ondoorzichtig (denk om de decimale punt). Halfdoorzichtig groen is:

```
rgba(0, 255, 0, 0.5)
```



Afbeelding 12.1 Drie RGB-tinten met in elke stap bij elke kleurwaarde 50 opgeteld. Het blijft blauwig, maar steeds lichter. Onder dezelfde tinten, maar met 0,5 doorzichtigheid/transparantie (12_01.html).

HSL en HSLA

In CSS3 is het kleurmodel HSL geïntroduceerd, ook bruikbaar met transparantie (HSLA). HSL staat voor *hue*, *saturation* en *lightness*, dat meestal wordt vertaald met tint, verzadiging en helderheid. Het is een model waarmee grafisch ontwerpers vaak vertrouwd zijn. Wellicht vindt u het intuïtiever dan RGB.

Het model is gebaseerd op het kleurenwiel. Een kleurenwiel loopt van 0 tot 360 graden van rood, naar groen, naar blauw terug naar rood:

- 0 – rood
- 60 – geel
- 120 – groen
- 180 – blauwgroen
- 240 – blauw
- 300 – roze
- 360 – rood

De CSS-notatie voor HSL bestaat uit drie waarden:

- het eerste getal is de tint: een getal van 0 tot 360;
- het tweede getal is het percentage verzadiging van 0 tot 100%;
- het derde getal is het percentage helderheid van 0 tot 100%.

Bekijk de volgende HSL-kleurdeclaraties:

- `hsl(0, 100%, 50%)` is rood;
- `hsl(360, 100%, 50%)` is ook rood;
- `hsl(120, 100%, 25%)` is donkergroen;
- `hsl(120, 100%, 75%)` is lichtgroen;
- `hsl(120, 50%, 75%)` is pastelgroen.



Altijd een procentteken

Ook als een van de procentuele waarden van een HSL-kleur 0 is, moet het procentteken worden geplaatst. Zonder procentteken is de declaratie ongeldig.

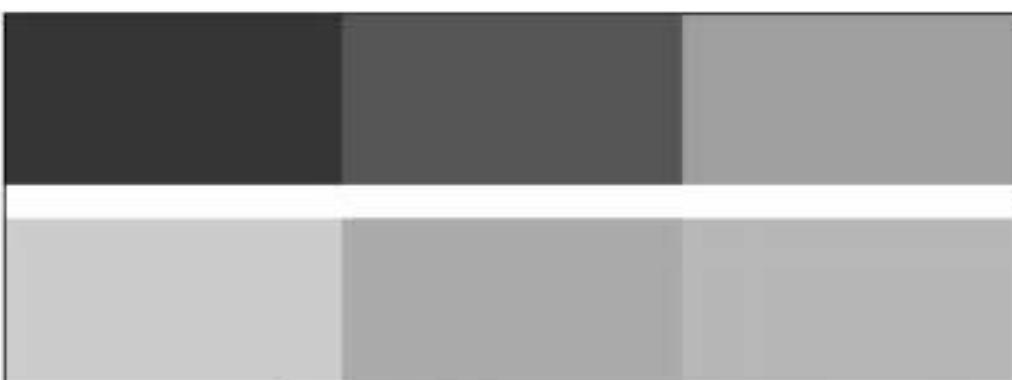
De waarden van een HSL-kleur doen dus het volgende:

- het eerste getal bepaalt de basiskleur;
- het tweede getal bepaalt hoe vol (verzadigd) die kleur is; 0% is geen kleur en 100% is de volle kleur;
- het derde getal bepaalt hoe donker of licht (helder) de kleur is. Een waarde van 0% is zwart en 100% is wit. De 'echte' kleur is 50%. Ter illustratie: RGB-rood (255, 0, 0) komt overeen met `hsl(0, 100%, 50%)`.

Voor bijvoorbeeld lichtgrijs is de kleurwaarde onbelangrijk, is de verzadiging 0% en de helderheid 90%:

```
hsl(0, 0%, 90%) /* lichtgrijs, kleur mag elke waarde hebben */
```

Om bij HSL transparantie toe te passen, gebruikt u de aanduiding `hsla` en een extra waarde voor de transparantie, net zoals bij `rgba`. Ook nu is 0 volledig



Afbeelding 12.2 Net zo'n reeks als in de voorgaande afbeelding, maar dan met HSL. De verzadiging en helderheid van kleurwaarde 120 zijn in stappen van 25% verhoogd. Onder loopt daarbij de transparantie in stappen van 25% op (12_01.html).

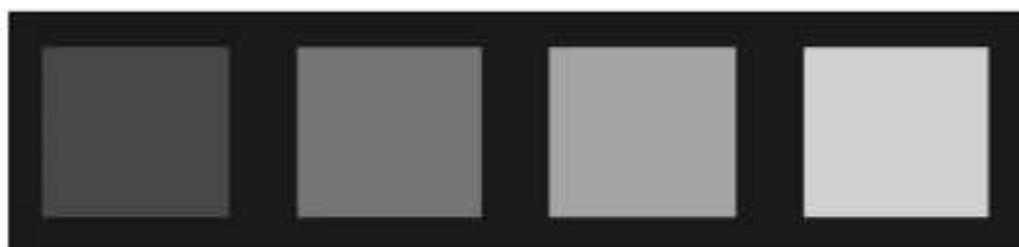
doorzichtig en geeft 1 de ondoorzichtige volledige kleur. Let op: de specificatie spreekt van opacity: ondoorzichtigheid. Vandaar dat opacity: 0 volledig doorzichtig is. Halfdoorzichtig donkergroen is:

```
hsla(120, 100%, 25%, 0.5)
```

Transparantie

Elk elementen dat een kleur kan hebben (tekst, een rand, een achtergrond enzovoort) kan als kleurwaarde het sleutelwoord transparent krijgen, een erfenis uit CSS 1. De kleur is dus volledig doorzichtig. De kleuren rgba(0, 0, 0, 0), hsla(0, 0%, 0%, 0) en transparent zijn dus allemaal transparant.

Behalve op de kleur kan transparantie ook op elementen worden ingesteld met de eigenschap opacity. Deze krijgt als waarde een getal van 0 (volledig transparant) tot 1 (geheel ondoorzichtig). De eigenschap kan worden gebruikt als alternatief voor visibility: hidden, waarbij ook nu het element ruimte blijft innemen. Stelt u deze eigenschap in, dan is het element onzichtbaar, ongeacht andere instellingen voor kleuren of achtergrondafbeeldingen.



Afbeelding 12.3 Een donkere achtergrondkleur met daarop blokken met een transparante witte achtergrond, geeft verschillend gekleurde blokken (12_01.html).



Kleuren in CSS3

Bekijk voor meer informatie en voorbeelden de kleurspecificatie van het W3C op www.w3.org/TR/css3-color/.

Kleur van tekst

De eigenschap color bepaalt de voorgrondkleur (tekstkleur) in een element. De kleur wordt aangegeven met een van de hiervoor genoemde methoden. Tekst is vaak zwart op een witte achtergrond, maar het paginabeeld kan net wat verfijnder zijn als de achtergrond (zie hierna) niet puur wit en de tekst niet puur zwart is. Houd wel de leesbaarheid in de gaten, die is altijd het belangrijkst. Experimenteer eens met andere waarden, zoals:

```
body {  
background-color: hsl(0, 0%, 95%);  
color: hsl(0, 0%, 5%)  
}
```

Een nauwelijks waarneembare achtergrondkleur die past bij het ontwerp kan natuurlijk ook; het hoeft niet grijs te zijn. Ook de tekstkleur kan worden afgestemd op de hoofdkleur van de site. Als dat een blauwtint is, werkt in plaats van zwart een heel donkere tint van dat blauw erg goed.

De achtergrond

Elke box die van de HTML-elementen wordt gemaakt heeft een achtergrond. Die ziet u gewoonlijk niet, want de standaardinstelling is transparant. Dankzij die standaardwaarde is de achtergrond van het ouderelement zichtbaar in de kindelementen. Met andere woorden: zolang u een achtergrond geen instelling geeft, is op elk niveau de achtergrond van het `<body>`-element zichtbaar. En daarover gesproken: alles wat hierna volgt over achtergronden kan ook worden toegepast op de achtergrond van de pagina: het element `<body>` dus.

Achtergrondkleur

De eigenschap `background-color` bepaalt de achtergrondkleur van een element. Dat kan variëren van de volledige achtergrond van de pagina (`<body>`) tot de achtergrond van een tekst of wat u maar bedenkt. Deze kleur komt achter een eventuele achtergrondafbeelding te staan. Daardoor kan de kleur als terugvalmechanisme worden gebruikt voor een ontbrekende afbeelding.

Achtergrondafbeelding

Op de achtergrond van elk element kan een afbeelding worden geplaatst. Sterker nog: er kunnen meerdere achtergrondafbeeldingen worden gebruikt. Al bij al kan het iets meer werk zijn dan een kleur, want er zijn zeven eigenschappen beschikbaar:

- `background-image`
- `background-repeat`
- `background-attachment`
- `background-position`
- `background-clip`
- `background-origin`
- `background-size`

background-image

Een achtergrondafbeelding plaatst u met `background-image`. Veel eenvoudiger kan het niet:

```
background-image: url('naam-afbeelding.jpg')
```

Het is verstandig om ook een achtergrondkleur in te stellen (in de tint van het beeld). Dan weet u zeker dat het contrast met de andere inhoud behouden blijft, ook als de afbeelding niet wordt getoond.



Kleurverloop als achtergrondafbeelding

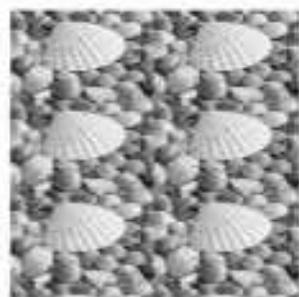
Een achtergrondafbeelding hoeft niet per se een bitmapafbeelding te zijn. Ook een kleurverloop kan als achtergrondafbeelding worden ingesteld. Zie de paragraaf *Kleurverloop als achtergrond*.

background-repeat

Een achtergrondafbeelding wordt standaard herhaald totdat alle ruimte in het blok is gevuld. U verandert dit met de eigenschap `background-repeat`. De volgende waarden zijn mogelijk:

- `repeat` zo vaak herhalen als nodig (standaard);
- `repeat-x` horizontaal herhalen;
- `repeat-y` verticaal herhalen;
- `no-repeat` niet herhalen;
- `space` zo vaak herhalen als past en de ruimte tussen de afbeeldingen gelijk verdelen;
- `round` zo vaak herhalen als past en de resterende ruimte vullen door de afbeelding te schalen.

Aan `background-repeat` mogen twee waarden worden gegeven. Dan geldt de eerste voor de horizontale richting en de tweede voor de verticale. Uiteraard worden `repeat-x` en `repeat-y` daarbij niet gebruikt.



Afbeelding 12.4 Een achtergrondafbeelding geplaatst met (v.l.n.r.) de standaardinstelling, `repeat-x` en `repeat-y` ([12_02.html](#)).

background-image

Een achtergrondafbeelding plaatst u met `background-image`. Veel eenvoudiger kan het niet:

```
background-image: url('naam-afbeelding.jpg')
```

Het is verstandig om ook een achtergrondkleur in te stellen (in de tint van het beeld). Dan weet u zeker dat het contrast met de andere inhoud behouden blijft, ook als de afbeelding niet wordt getoond.



Kleurverloop als achtergrondafbeelding

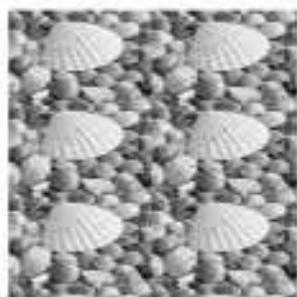
Een achtergrondafbeelding hoeft niet per se een bitmapafbeelding te zijn. Ook een kleurverloop kan als achtergrondafbeelding worden ingesteld. Zie de paragraaf *Kleurverloop als achtergrond*.

background-repeat

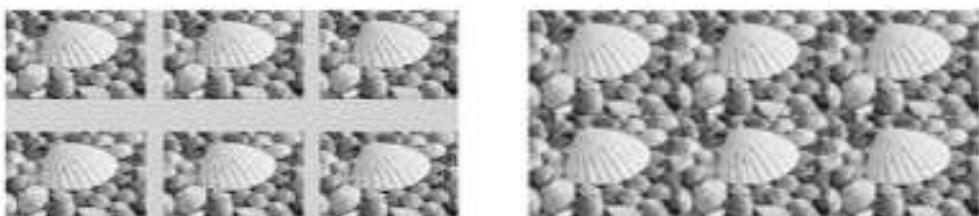
Een achtergrondafbeelding wordt standaard herhaald totdat alle ruimte in het blok is gevuld. U verandert dit met de eigenschap `background-repeat`. De volgende waarden zijn mogelijk:

- `repeat` zo vaak herhalen als nodig (standaard);
- `repeat-x` horizontaal herhalen;
- `repeat-y` verticaal herhalen;
- `no-repeat` niet herhalen;
- `space` zo vaak herhalen als past en de ruimte tussen de afbeeldingen gelijk verdelen;
- `round` zo vaak herhalen als past en de resterende ruimte vullen door de afbeelding te schalen.

Aan `background-repeat` mogen twee waarden worden gegeven. Dan geldt de eerste voor de horizontale richting en de tweede voor de verticale. Uiteraard worden `repeat-x` en `repeat-y` daarbij niet gebruikt.



Afbeelding 12.4 Een achtergrondafbeelding geplaatst met (v.l.n.r.) de standaardinstelling, `repeat-x` en `repeat-y` ([12_02.html](#)).



Afbeelding 12.5 Het effect van space en round (12_02.html).

Een afbeelding alleen horizontaal herhalen kan zo op twee manieren worden aangegeven:

```
background-repeat: repeat-x  
background-repeat: repeat no-repeat
```



Verticale lijn

Met een kleine afbeelding van bijvoorbeeld 5 bij 5 pixels maakt u in een handomdraai een rechtopstaande lijn in de pagina. U hoeft alleen de afbeelding verticaal te herhalen: `background: url('lijn.jpg') repeat-y`. Dit is een oldschooltechniek, maar wie weet komt die nog eens van pas.

background-attachment

Een achtergrondafbeelding schuift standaard mee met de rest van de pagina, maar u kunt hem ook vastzetten. Daarvoor is de eigenschap `background-attachment`. Deze heeft de waarden:

- scroll (standaard)
- local (meeschuiven met de inhoud van een element als dat een schuifbalk heeft)
- fixed (vastgepind aan het browservenster).

background-position

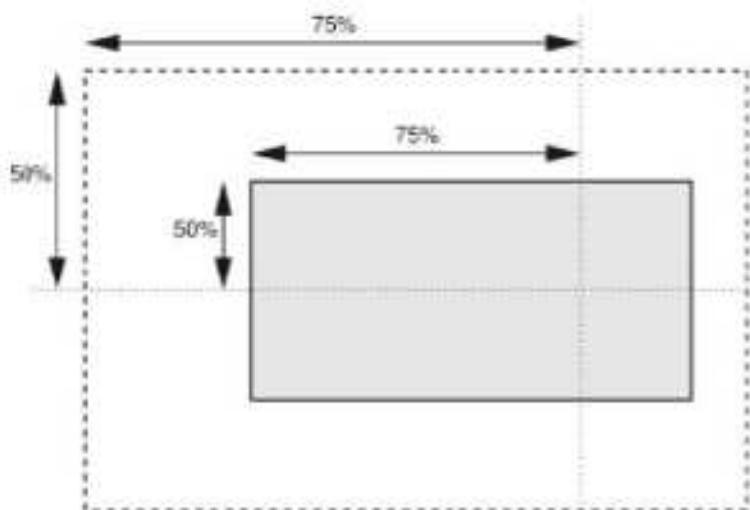
Een achtergrondafbeelding kan elke positie in het venster (de container) krijgen. Standaard komt deze tegen de rand vanuit het punt linksboven: dat is positie 0% 0%. Plaatsing regelt u met `background-position`. Dit is zo'n eigenschap waarbij u vanwege de vele mogelijkheden gemakkelijk de draad kwijttraakt (zie ook www.w3.org/TR/css-backgrounds-3/#the-background-position).

De eigenschap accepteert sleutelwoorden (`left, center, right, top, bottom`), een lengte of een percentage. Er kunnen één tot vier waarden zijn. Als maar één

waarde wordt gegeven, is de andere automatisch center. Gebruikt u vier waarden, dan zijn dat combinaties van een sleutelwoord en een lengte of percentage.

Een percentage is altijd relatief ten opzichte van de box en de afbeelding. Dat wil zeggen: een horizontale verschuiving van 75% plaatst het punt op 75% van de breedte van de afbeelding op 75% van de linkerkant van de box. Het resultaat uit de afbeelding is hetzelfde als:

`background-position: left 75% top 50%`



Afbeelding 12.6 Het resultaat van `background-position: 75% 50%` (bron: W3C).

Bij een lengte gaat het om een verschuiving vanuit de sleutelwoordpositie (of als die niet is gegeven vanuit de hoek linksboven). De volgende declaratie zet de hoek rechtsonder van afbeelding 50px vanuit de hoek rechtsonder van de box:

`background-position: right 50px bottom 50px`

De declaratie hierna zet het beeld op de positie 50px omlaag en naar rechts vanuit de hoek linksboven van de box:

`background-position: 50px 50px`

Deze declaratie zet het beeld op de positie 50px naar rechts en verticaal gecentreerd (want de weggelaten waarde van top wordt center):

`background-position: 50px`



Pictogrammen uitknippen: sprites

Het kan handig zijn om pictogrammen voor een webpagina te verzamelen in één afbeeldingbestand. Vervolgens kan met background-position een deel uit dat afbeeldingbestand worden getoond: het gewenste pictogram. Het afbeeldingbestand hoeft maar één keer te worden gedownload. Dat scheelt kB's en tijd, zeker als het veel pictogrammen zijn. Een voorbeeld is een muisaanwijseffect met twee afbeeldingen. Voor de standaardweergave wordt gewoon de achtergrondafbeelding gedownload in een container met de afmetingen van de sprite. De background-position is standaard linksboven (0,0). Voor het aanwijseffect hoeft alleen de background-position te worden verplaatst naar de tweede sprite. In dit voorbeeld staat die 100px naar rechts.

```
.sprite {
    width: 100px;
    height: 70px;
    background-image: url('../images/sprite.png');
}

.sprite:hover {
    background-position: 100px 0;
}
```



Afbeelding 12.7 De afbeelding met de twee sprites (rechts is in kleur...) (12_02.html).

background-clip

De eigenschap `background-clip` bepaalt tot in welk gebied van de box de overloop van de achtergrondafbeelding wordt getoond. De afbeelding moet dus groot genoeg zijn (of worden herhaald) om dat gebied te kunnen bedekken. De mogelijkheden zijn:

- `border-box` de buitenkant van de rand (standaard);
- `padding-box` de buitenkant van de padding;
- `content-box` de buitenkant van de inhoud.

De afbeelding loopt bij de standaardinstelling border-box dus door achter de rand en kan zo worden gebruikt voor transparantie-effecten. Bedenk dat het hier gaat om de *overloop* van het beeld. Dat betekent dat een achtergrondafbeelding die ‘normaal’ wordt geplaatst – vanuit de hoek linksboven – in elk geval bovenaan en links nooit overloopt. Ook bij background-size: contain is er nooit overloop. Als de achtergrond mag worden herhaald (background-repeat) dan loopt de herhaling vanzelf door onder rand.

Is het de bedoeling om de overloop als effect te gebruiken, bijvoorbeeld achter een halftransparante rand, dan kan de (voldoende grote) afbeelding of worden gecentreerd (background-position: center) of zelfs precies worden gepositioneerd met verschuivingswaarden.



Afbeelding 12.8 Twee precies passende afbeeldingen zonder overloop, maar rechts is repeat toegestaan waardoor overloop wordt geforceerd onder de rand (12_02.html).

background-origin

Met background-position wordt een achtergrondafbeelding verschoven om precies het gewenste fragment van een foto als achtergrond in beeld te krijgen. Wanneer het er alleen om gaat dat de achtergrond wordt getekend vanaf de buitenkant van de rand, kan het een stuk eenvoudiger: verplaats de oorsprong van het beeld met background-origin.



Afbeelding 12.9 Het effect van border-box, padding-box en content-box (v.l.n.r.). De boxen hebben 32px padding (12_02.html).

Er zijn weer drie mogelijkheden:

- border-box plaats vanuit de buitenkant rand;
- padding-box plaats vanuit de binnenkant rand (standaard);
- content-box plaats vanuit de binnenkant padding.

background-size

Een achtergrondafbeelding wordt standaard getoond met de eigen afmetingen. Wanneer het beeld groter of kleiner is dan het achtergrondgebied, kan de afbeelding worden geschaald met `background-size`. Deze eigenschap accepteert als waarden sleutelwoorden, lengten en percentages. Bij twee waarden voor een lengte of percentage is de eerste voor de breedte en de tweede voor de hoogte. Bij één waarde is de tweede auto.

- auto de eigen afmetingen van het beeld (standaard);
- contain schaalt de afbeelding zodat deze volledig wordt getoond en behoudt de beeldverhouding. Daardoor kan een deel van het achtergrondgebied onbedekt blijven;
- cover schaalt de afbeelding zodat het hele achtergrondgebied wordt bedekt. De beeldverhouding wordt behouden en de afbeelding wordt bijgesneden als dat nodig is.



Afbeelding 12.10 Het verschil tussen `background-size: auto`, `contain` en `cover` (v.l.n.r.). Bij `contain` blijft er gebied onbedekt, doordat de box een andere verhouding heeft dan de achtergrondafbeelding. De achtergrond wordt niet herhaald.



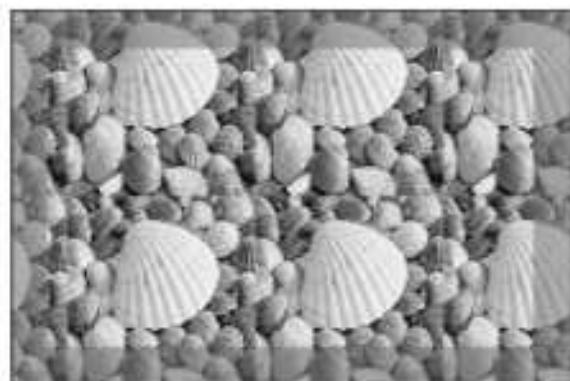
Meer voorbeelden

Op www.w3.org/TR/css-backgrounds-3/#the-background-size zijn voorbeelden te vinden van het werken met lengten en percentages.

background

Voor de eigenschappen van de achtergrond is ook een korte notatie: background. Bedenk dat kort niet per se hetzelfde is als duidelijk; vaak is het overzichtelijker om de afzonderlijke eigenschappen te gebruiken, zeker als veel eigenschappen moeten worden ingesteld. Eigenschappen die niet in de korte notatie worden opgenomen, krijgen hun initiële waarde.

Achtereenvolgens worden aangegeven background-image, background-position, background-size, background-repeat, background-origin, background-clip en background-attachment. Bij de laatste afbeelding (het mogen er meer zijn, zie de volgende paragraaf) kan ook background-color worden gebruikt (die zichtbaar is als de afbeelding niet kan worden geladen). Als position en size worden ingesteld, worden hun waarden gescheiden door een slash (/).



Afbeelding 12.11 Het resultaat van de code, maar dat hangt natuurlijk volledig af van de gebruikte afbeelding. Dit geeft overigens een heel mooi scroll-effect (12_02.html).

Een voorbeeld:

```
div {  
    background: url('../images/schelp_200.png') 40% / 10em 10em hs1(200, 75%, 50%) round fixed  
    border-box;  
}
```

Dit is hetzelfde als:

```
div {  
    background-color: hs1(200, 75%, 50%);  
    background-position: 40% 50%;  
    background-size: 10em 10em;  
    background-repeat: round;  
    background-clip: border-box;
```

```
background-origin: border-box;
background-attachment: fixed;
background-image: url('achtergrond.png')
}
```

Verschillende achtergrondafbeeldingen

Een blok kan verschillende achtergrondafbeeldingen hebben, meerdere tegelijk dus. Alles wat daar voor nodig is, is een uitbreiding van de declaratie (en geschikte afbeeldingen natuurlijk). De extra declaraties worden met een komma van elkaar gescheiden. Dat geldt ook voor de korte notatie.

De volgende declaratie plaatst twee afbeeldingen op de achtergrond, een linksboven en een rechtsboven:

```
{ background:
url(plaatje1.jpg) top left no-repeat,
url(plaatje2.jpg) top right no-repeat;
}
```

Op vergelijkbare wijze kan in elke hoek een afbeelding worden geplaatst, of bijvoorbeeld links en rechts in het midden:

```
{ background:
url(plaatje1.jpg) center left no-repeat,
url(plaatje2.jpg) center right no-repeat;
}
```

De afbeeldingen worden gestapeld en de laatstgenoemde komt onderop. Hiervan kunt u gebruikmaken om een achtergrond in het hele blok te plaatsen, met accentafbeeldingen in de hoeken. (Een verfijnd resultaat wordt bereikt met 'opengewerkte'/transparante hoekafbeeldingen, waar de achtergrond doorheen komt.) Neem in de declaratie eerst de hoekafbeeldingen op en als laatste de totale achtergrond.

Let op: als ook een achtergrondkleur wordt gebruikt, moet die waarde komen in de declaratie van de laatste afbeelding. Doet u dat niet, dan bedekt de achtergrondkleur de andere afbeeldingen. Hoewel u alle kenmerken per afbeelding kwijt kunt in de korte notatie, kan het overzichtelijker zijn om de afzonderlijke eigenschappen te gebruiken:

```
.m-background {
margin-top: 3em;
width: 600px;
```

```
height: 400px;  
background-image: url('../images/CSS3_logo.png'),  
url('../images/tim-mossholder-1362421-unsplash.jpg');  
background-position: bottom 20% right 30%, center center;  
background-color: white;  
background-repeat: no-repeat;  
}
```



Afbeelding 12.12 Twee achtergrondafbeeldingen (12_02.html).

Kleurverloop als achtergrond

Een *image* is in CSS3 een vrij ruim begrip en dat geeft mede meer mogelijkheden aan de eigenschap `background-image`. Een kleurverloop als achtergrond is namelijk ook mogelijk. Het maken van kleurverlopen wordt uitgelegd in de volgende paragraaf. Als achtergrondafbeelding is de algemene declaratie:

```
background-image: linear-gradient(kleur, kleur);
```

Alle moderne browsers ondersteunen deze eigenschap en een terugvalkleur is geen noodzaak, maar met terugvalkleur wordt de declaratie bijvoorbeeld:

```
background: linear-gradient(red, yellow) orange;
```

Kleurverlopen

Dankzij CSS3 zijn kleurverlopen mogelijk. Een kleurverloop (in het Engels *gradient*) is een vloeiend verloop van de ene kleur naar een andere. Het is een van de nieuwe *functies* in CSS. Het verschil tussen een functie en een eigenschap is dat een functie de opgeven waarden verwerkt tot een nieuw resultaat, terwijl een eigenschap gewoon die waarde krijgt toegewezen.

Terug naar het kleurverloop. CSS ondersteunt twee typen verloop:

- `linear-gradient()` een rechtlijnig horizontaal, verticaal of diagonaal verloop;
- `radial-gradient()` een rond verloop langs de straal van een cirkel.

Lineair verloop

Een lineair verloop volgt een rechte lijn. Die lijn kan in alle richtingen lopen. Er zijn minimaal twee kleuren nodig voor een verloop, maar het kunnen er ook meer zijn. Het punt waarop de ene kleur stopt en een andere begint is een kleurstop.

Standaard worden de kleuren evenredig verdeeld over de lengte van het verloop, maar met een kleurstopwaarde kan dat worden veranderd. Een stop kan in een lengte of een percentage worden aangegeven, waarbij de totale lengte van het verloop 100% is. Vanzelfsprekend wordt een stop aangegeven bij de kleur waarop die betrekking heeft. Zie ook het voorbeeld hierna.

Deze volgende waarden kunnen worden ingesteld, gescheiden door een komma:

- de richting in graden (eenheid deg) of met sleutelwoorden:
 - `to top` gelijk aan 0deg
 - `to right` gelijk aan 90deg
 - `to bottom` (standaard) gelijk aan 180deg
 - `to left` gelijk aan 270deg
- de kleurwaarde, optioneel gevolgd door een kleurstop in een lengte of een percentage.

De volgende declaratie maakt een standaardverloop van boven naar beneden:

```
linear-gradient(black, white);
```

Kleurverlopen

Dankzij CSS3 zijn kleurverlopen mogelijk. Een kleurverloop (in het Engels *gradient*) is een vloeiend verloop van de ene kleur naar een andere. Het is een van de nieuwe *functies* in CSS. Het verschil tussen een functie en een eigenschap is dat een functie de opgeven waarden verwerkt tot een nieuw resultaat, terwijl een eigenschap gewoon die waarde krijgt toegewezen.

Terug naar het kleurverloop. CSS ondersteunt twee typen verloop:

- `linear-gradient()` een rechtlijnig horizontaal, verticaal of diagonaal verloop;
- `radial-gradient()` een rond verloop langs de straal van een cirkel.

Lineair verloop

Een lineair verloop volgt een rechte lijn. Die lijn kan in alle richtingen lopen. Er zijn minimaal twee kleuren nodig voor een verloop, maar het kunnen er ook meer zijn. Het punt waarop de ene kleur stopt en een andere begint is een kleurstop.

Standaard worden de kleuren evenredig verdeeld over de lengte van het verloop, maar met een kleurstopwaarde kan dat worden veranderd. Een stop kan in een lengte of een percentage worden aangegeven, waarbij de totale lengte van het verloop 100% is. Vanzelfsprekend wordt een stop aangegeven bij de kleur waarop die betrekking heeft. Zie ook het voorbeeld hierna.

Deze volgende waarden kunnen worden ingesteld, gescheiden door een komma:

- de richting in graden (eenheid deg) of met sleutelwoorden:
 - `to top` gelijk aan 0deg
 - `to right` gelijk aan 90deg
 - `to bottom` (standaard) gelijk aan 180deg
 - `to left` gelijk aan 270deg
- de kleurwaarde, optioneel gevolgd door een kleurstop in een lengte of een percentage.

De volgende declaratie maakt een standaardverloop van boven naar beneden:

```
linear-gradient(black, white);
```

```
background-image: linear-gradient(to right, white 15px, black 15px);
background-size: 25px 1px; /* 25px is de breedte en 1px de hoogte van het verloop*/
background-repeat: repeat; /* door de herhaling van het patroon ontstaan de 'lamellen' */
```

Radiaal verloop

Een radiaal kleurverloop volgt de straal van een cirkel. Vanuit een middelpunt verloopt de ene kleur cirkelvormig in de andere. Dat middelpunt is standaard het middelpunt van het blok waarin het verloop wordt toegepast. Dat punt kan worden aangepast met dezelfde waarden die `background-position` gebruikt: de sleutelwoorden `left`, `center`, `right`, `top`, `bottom`, een lengte of een percentage, voorafgegaan door het woord `at`. Bij dit verloop kan ook de vorm worden ingesteld: `circle` of `ellipse` (afgeplatte cirkel).

De grootte van het verloop kan worden bepaald met een lengte voor de straal. Alleen bij `ellipse` kan ook een percentage worden gebruikt en zijn er twee waarden: voor de horizontale en de verticale straal.

Voor de grootte zijn sleutelwaarden beschikbaar die vanuit het middelpunt het eindpunt aangeven (hoewel de declaratie andersom wordt geformuleerd):

- `closest-side` dichtstbijzijnde kant;
- `farthest-side` verst verwijderde kant;
- `closest-corner` dichtstbijzijnde hoek;
- `farthest-corner` verst verwijderde hoek.

Gewoon proberen wat er gebeurt bij bepaalde instellingen is zeker bij radiale verlopen een prima manier om de werking te doorgronden. Wat voorbeelden.

Een cirkelvormig verloop:

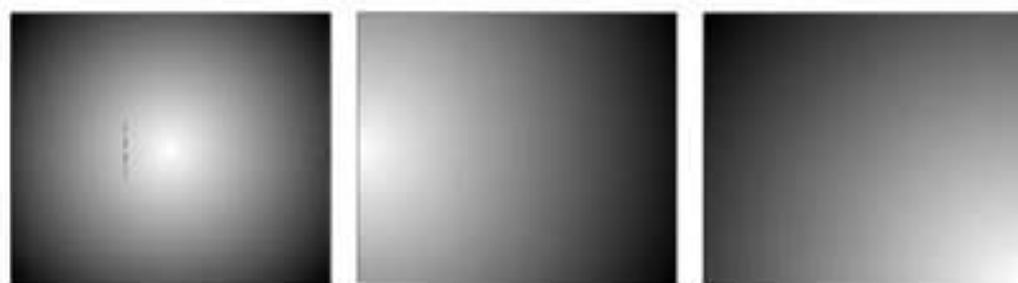
```
radial-gradient(circle, white, black);
```

Een verloop dat begint op positie midden links:

```
radial-gradient(circle at left, white, black);
```

Een verloop vanuit rechtsonder naar de hoek linksboven:

```
radial-gradient(farthest-corner circle at right bottom, white, black);
```



Afbeelding 12.14 Drie radiale verlopen: cirkelvormig, vanuit midden links en vanuit rechts onder (12_03.html).

Repeterende verlopen

Het lineaire en het radiale verloop hebben elk een variant waarmee het patroon wordt herhaald, wat spectaculaire resultaten kan opleveren. De definitie van het verloop is bijna hetzelfde als bij de enkelvoudige verlopen, met als belangrijkste verschil dat de laatste kleur een stopwaarde moet hebben. Vanaf dat punt wordt het patroon herhaald. De naam van de functies is `repeating-linear-gradient()` en `repeating-radial-gradient()`. Met lengtewaarden (pixels) is het patroon aanmerkelijke eenvoudiger te controleren dan met procenten:

```
repeating-linear-gradient(black, grey, white 50px);
```

De kleurstop `white 50px` betekent nu: op 50px vanaf het begin van het patroon moet de kleur wit zijn, daarna wordt het patroon herhaald. De laatste kleurwaarde markeert dus het eind van het patroon.

Nu wordt het patroon uitgebreid met een stop grijs. Tel voor de waarde ervan de gewenste lengte van de kleur op bij de laatste stopwaarde:

```
repeating-linear-gradient(black, grey, white 50px, grey 70px);
```

Het repeterende verloop uit de paragraaf *Lineair verloop* kan dus ook als volgt worden gemaakt. Belangrijk hierbij is dat een startkleur wordt gedefinieerd, anders werkt het niet:

```
background-image: repeating-linear-gradient(  
    to right,  
    white,  
    white 15px,  
    black 15px,  
    black 25px);
```



Afbeelding 12.15 Herhaalde kleurverlopen. Van het verloop rechts is het resultaat hetzelfde als in afbeelding 12.13 (12_03.html).

Afgeronde hoeken

Afgeronde hoeken waren ooit net zo'n moeizaam onderwerp als verticale uitlijning. Sinds CSS3 zijn er kant-en-klare ronde hoeken, te vinden in de CSS3-module *Backgrounds and Borders level 3*.

Elke hoek kan afzonderlijk worden ingesteld met de vier eigenschappen:

- border-top-left-radius
- border-top-right-radius
- border-bottom-right-radius
- border-bottom-left-radius.

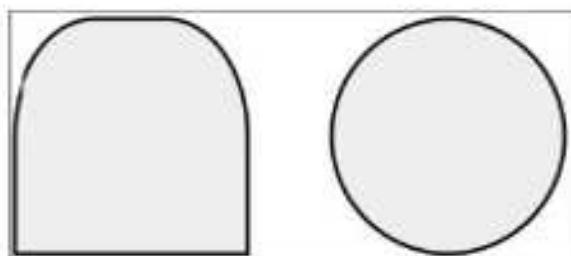
Per hoek kunnen een of twee waarde worden gegeven; bij twee waarden geldt de eerste voor horizontale zijde en de tweede voor de verticale zijde. Op deze manier zijn onregelmatige hoeken mogelijk.

Mogelijke waarden zijn een lengte (px, em) of een percentage. Het percentage werkt handig omdat het is gerelateerd aan de lengte van de box. Een zijde van 400px met hoek van 25% begint op 100px uit de hoek. Op een vierkant blok geeft 50% op alle hoeken dus een cirkel.

Op een vierkant blok worden de volgende hoeken ingesteld:

```
.round1 {
    border-top-left-radius: 35% 50%;
    border-top-right-radius: 35% 50%;
}

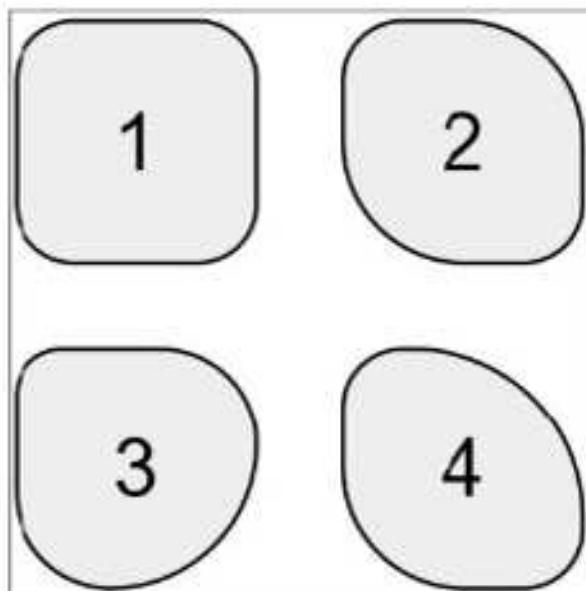
.round2 {
    border-radius: 50%;
```



Afbeelding 12.16 Het resultaat van de code: een brood en een cirkel (12_04.html).

De cirkel is gemaakt met de verkorte eigenschap border-radius. Deze kan één, twee, drie of vier waarden hebben. Eén waarde geldt voor alle hoeken. Bij twee waarden is linksboven hetzelfde als rechtsonder en is rechtsboven hetzelfde als linksonder. Bij drie waarden is de eerste voor linksboven, de tweede voor rechtsboven en linksonder, en de derde voor rechtsonder. Bij vier waarden is de eerste voor de hoek linksboven en dan met de klok mee.

```
.round1 {  
    border-radius: 25%;  
}  
.round2 {  
    border-radius: 25% 50%;  
}  
.round3 {  
    border-radius: 25% 50% 75%;  
}
```



Afbeelding 12.17 Hoeken gemaakt met één, twee, drie en vier waarden voor border-radius (12_04.html).

```
.round4 {
    border-radius: 25% 70% 25% 50%;
}
```

Ook met border-radius kunnen asymmetrische hoeken worden gemaakt. Breid daarvoor de declaratie uit met een slash en een tweede reeks van één tot vier waarden. Voor de tweede reeks geldt dezelfde toewijzingsformule en het hoeft net hetzelfde aantal te zijn als voor de slash.

```
.round7 {
    border-radius: 25% 50% / 25% 70% 25% 50%;
}
```

Het zal u zijn opgevallen dat de achtergrond binnen de hoeklijntjes wordt gekleurd en dat gebeurt ook als er geen rand is ingesteld met border. Bij elk blok dat afgeronde hoeken krijgt, wordt de achtergrond van het blok weggeknipt volgens de ingestelde rondering. De box houdt wel zijn oorspronkelijke buitenmaten.



Afbeelding 12.18 Geen rand, wel bijgeknipt, maar de box blijft even groot (12_04.html).

De mogelijkheden met hoeken zijn groot, omdat de vorm niet alleen afhangt van de ingestelde hoek, maar ook van de vorm van de box. Bovendien kunnen verschillende diktes en soorten randen worden ingesteld op afzonderlijke zijden. U kunt zomaar een avond stukslaan met experimenteren.

Randafbeelding

CSS3 introduceert nog een nieuw randverschijnsel: border-image. Hierbij wordt een afbeelding gebruikt voor de rand van een element. Zo kan de randstijl (border-style) worden vervangen door een afbeelding die kan worden gesegmenteerd, geschaald en uitgerekt om in de rand te passen. Om border-image te kunnen gebruiken, moet zowel border-width als border-style zijn ingesteld, waarbij de randstijl fungert als terugvalmechanisme wanneer de afbeelding niet kan worden geladen (of de browser de eigenschap niet onder-

```
.round4 {
    border-radius: 25% 70% 25% 50%;
}
```

Ook met border-radius kunnen asymmetrische hoeken worden gemaakt. Breid daarvoor de declaratie uit met een slash en een tweede reeks van één tot vier waarden. Voor de tweede reeks geldt dezelfde toewijzingsformule en het hoeft net hetzelfde aantal te zijn als voor de slash.

```
.round7 {
    border-radius: 25% 50% / 25% 70% 25% 50%;
}
```

Het zal u zijn opgevallen dat de achtergrond binnen de hoeklijntjes wordt gekleurd en dat gebeurt ook als er geen rand is ingesteld met border. Bij elk blok dat afgeronde hoeken krijgt, wordt de achtergrond van het blok weggeknipt volgens de ingestelde rondering. De box houdt wel zijn oorspronkelijke buitenmaten.



Afbeelding 12.18 Geen rand, wel bijgeknipt, maar de box blijft even groot (12_04.html).

De mogelijkheden met hoeken zijn groot, omdat de vorm niet alleen afhangt van de ingestelde hoek, maar ook van de vorm van de box. Bovendien kunnen verschillende diktes en soorten randen worden ingesteld op afzonderlijke zijden. U kunt zomaar een avond stukslaan met experimenteren.

Randafbeelding

CSS3 introduceert nog een nieuw randverschijnsel: border-image. Hierbij wordt een afbeelding gebruikt voor de rand van een element. Zo kan de randstijl (border-style) worden vervangen door een afbeelding die kan worden gesegmenteerd, geschaald en uitgerekt om in de rand te passen. Om border-image te kunnen gebruiken, moet zowel border-width als border-style zijn ingesteld, waarbij de randstijl fungert als terugvalmechanisme wanneer de afbeelding niet kan worden geladen (of de browser de eigenschap niet onder-

steunt). De randafbeelding beïnvloedt niet de ruimte die een blok inneemt, dat doen alleen de breedte en de stijl van de rand. Met andere woorden: de afmeting van de randafbeelding wordt aangepast aan de randdikte, niet andersom.

border-image-source

De basis wordt gevormd door de bronafbeelding. Deze wordt geladen met:

```
border-image-source: url ('bronafbeelding.png')
```

In dit voorbeeld wordt een bitmap geladen, maar er kunnen ook kleurverlopen worden toegepast.

border-image-slice

Vervolgens wordt de afbeelding in moten gesneden en die stukken (*slices*) worden gebruikt voor de hoeken en zijden van de rand. De eigenschap is border-image-slice. De waarde kan bestaan uit één tot vier lengtewaarden of één tot vier percentages, en optioneel het sleutelwoord `fill`.

- De lengtewaarde(n) geeft aan hoe dik (px, em, rem) de plak is die uit de bronafbeelding wordt gesneden. De volgorde is boven, rechts, onder, links (net als bij andere eigenschappen dus. Eén getal geldt voor alle kanten, twee getallen voor boven/onder en rechts/links, en drie getallen voor boven, rechts/links en onder).
- Voor het percentage of de percentages gaat het om een procentueel deel van de bronafbeelding, in dezelfde volgorde als bij getallen.
- Met de optionele waarde `fill` wordt ook het middengebied van de bronafbeelding gebruikt in het element. De afmetingen kunnen niet worden ingesteld. Als het formaat van het element en de bronafbeelding niet overeenkomt, wordt de afbeelding vervormd.

border-image-width

De breedte van de randafbeelding wordt ingesteld met `border-image-width`. De eigenschap accepteert één tot vier waarden (voor boven, rechts, onder, links; invulling ontbrekende waarden zoals bij de `margin`): lengte, percentage, een getal of `auto`. Het getal staat voor het aantal keren waarmee de `border-width` wordt vermenigvuldigd. Het sleutelwoord `auto` gebruikt de afmetingen van de slices of als dat niet kan de `border-width`. De startwaarde van `border-image-width` is 1, wat gelijk staat aan de `border-width`.

border-image-outset

De randafbeelding begint standaard op de buitengrens van de rand en gaat dan naar binnen. Dit kan worden omgedraaid met de eigenschap border-image-outset. Ook deze eigenschap accepteert de vier bekende waarden als lengte (px, em, rem) of een nummer (aantal keren border-width). De waarde geeft aan hoeveel de randafbeelding uitsteekt. Eén waarde geldt voor alle randen, twee waarden voor verticaal en horizontaal, drie waarden voor boven, horizontaal en onder, vier waarden voor boven, rechts, onder, links.

Houd er rekening mee dat de ruimte voor het totale blok hiermee toeneemt. Aangrenzende inhoud kan worden overlapt of de randafbeelding kan worden afgekapt als het ouderelement geen ruimte meer heeft.

border-image-repeat

De laatste eigenschap bepaalt hoe de beeldsegmenten (slices) worden toegepast om de ruimte van de rand te vullen. De eigenschap is border-image-repeat. Er kunnen twee waarden worden gegeven, waarbij de eerste voor de horizontale randen geldt en de tweede voor de verticale. Mogelijke waarden zijn:

- stretch de afbeelding wordt uitgerekt (standaard)
- repeat de afbeelding wordt herhaald;
- round de afbeelding wordt zo vaak herhaald als nodig om de rand te vullen met een geheel aantal vormen, zo nodig wordt het beeld geschaald;
- space herhalen zoals bij round, maar resterende witruimte wordt verdeeld.

Met de verkorte notatie border-image worden alle instellingen ineens gedaan, in de volgorde -source, -slice, -width, -outset, -repeat. Ontbrekende waarden krijgen de startwaarde.

Voorbeeld van een randafbeelding

Zo veel theorie, dat vraagt om een voorbeeld (het is ook te vinden op www.w3.org/TR/css3-background/#border-images). De basis wordt gevormd door een afbeelding van 81 bij 81 pixels, die in negen gelijke stukken moet worden verdeeld. De achtergrond is transparant.



Afbeelding 12.19 Links de bronafbeelding in zijn content-box, in het midden de rand en rechts de rand met afbeelding (12_05.html).

```
border: 16px double hsl(0, 100%, 75%);  
border-image-source: url('border-image.png');  
border-image-slice: 27;  
border-image-repeat: round stretch;
```

- Eerst wordt een dubbele (border-style), rozige rand in van 16px (border-width) ingesteld.
- Daarna wordt de afbeelding voor de rand ingesteld.
- Met border-image-slice wordt de dikte van de plak ingesteld: 27px. De afbeelding moet immers in drieën worden gedeeld en is 81 pixels breed ($3 \times 27 = 81$).
- De laatste regel bepaalt dat de afbeelding horizontaal wordt herhaald en verticaal wordt uitgerekt.

In de verkorte notatie ziet dezelfde declaratie er als volgt uit:

```
border-image: url('border-image.png') 27 round stretch;
```



Formaat van de afbeelding

In dit voorbeeld is de afbeelding 81 pixels hoog en breed. De negen segmenten zijn daardoor vierkanten van 27 x 27 pixels ($3 \times 27 = 81$), waarin precies één ruit past. In principe is elke afmeting mogelijk, maar het is wel verstandig een bitmap te maken waarbij de afmetingen van het patroon enigszins passen bij de rand en het formaat van het kader. De afbeelding kan worden geschaald en sterk vergrootte bitmaps kunnen rafelig worden. Verkleinen leidt minder snel tot vervormd beeld. U kunt natuurlijk ook een vectorafbeelding gebruiken. SVG's worden nooit blokkerig.

Schaduw

Net als afgeronde hoeken en randafbeeldingen is ook schaduw een CSS3-toevoeging. Met de eigenschap box-shadow wordt een element in een handomdraai van een schaduw voorzien. De algemene vorm is:

```
box-shadow: x-verschuiving y-verschuiving vervagingsstraal verspreiding kleur (inset);
```

X is de verschuiving naar rechts en y is de verschuiving omlaag (bij negatieve waarden naar links en omhoog). De *vervagingsstraal* is de afstand waarin de schaduw verloopt van volle kleur naar transparant. De *verspreiding* vergroot deze afstand en maakt daarmee de schaduw zachter. Kleur kan elk van de

```
border: 16px double hsl(0, 100%, 75%);  
border-image-source: url('border-image.png');  
border-image-slice: 27;  
border-image-repeat: round stretch;
```

- Eerst wordt een dubbele (border-style), rozige rand in van 16px (border-width) ingesteld.
- Daarna wordt de afbeelding voor de rand ingesteld.
- Met border-image-slice wordt de dikte van de plak ingesteld: 27px. De afbeelding moet immers in drieën worden gedeeld en is 81 pixels breed ($3 \times 27 = 81$).
- De laatste regel bepaalt dat de afbeelding horizontaal wordt herhaald en verticaal wordt uitgerekt.

In de verkorte notatie ziet dezelfde declaratie er als volgt uit:

```
border-image: url('border-image.png') 27 round stretch;
```



Formaat van de afbeelding

In dit voorbeeld is de afbeelding 81 pixels hoog en breed. De negen segmenten zijn daardoor vierkanten van 27 x 27 pixels ($3 \times 27 = 81$), waarin precies één ruit past. In principe is elke afmeting mogelijk, maar het is wel verstandig een bitmap te maken waarbij de afmetingen van het patroon enigszins passen bij de rand en het formaat van het kader. De afbeelding kan worden geschaald en sterk vergrootte bitmaps kunnen rafelig worden. Verkleinen leidt minder snel tot vervormd beeld. U kunt natuurlijk ook een vectorafbeelding gebruiken. SVG's worden nooit blokkerig.

Schaduw

Net als afgeronde hoeken en randafbeeldingen is ook schaduw een CSS3-toevoeging. Met de eigenschap box-shadow wordt een element in een handomdraai van een schaduw voorzien. De algemene vorm is:

```
box-shadow: x-verschuiving y-verschuiving vervagingsstraal verspreiding kleur (inset);
```

X is de verschuiving naar rechts en y is de verschuiving omlaag (bij negatieve waarden naar links en omhoog). De *vervagingsstraal* is de afstand waarin de schaduw verloopt van volle kleur naar transparant. De *verspreiding* vergroot deze afstand en maakt daarmee de schaduw zachter. Kleur kan elk van de

```
}

.shadow3 {
    box-shadow: 0 0 0 10px rgba(0,0,0,0.5);
}
```



Afbeelding 12.21 Het resultaat van verschillende instellingen voor `box-shadow`. Alleen de middelste heeft vervaging (12_05.html).

Material design

Webdesign is onderhevig aan mode en trends. Plat en strak (Windows-tegels) ontwikkelt zich naar meer levendigheid in het beeld door er diepte en beweging aan te geven. Google heeft daarvoor *material design* ontwikkeld. Material design is een visuele taal gebaseerd op papier en inkt en maakt gebruik van responsieve animaties en overgangen, padding en diepte-effecten met licht en schaduw. Beweging is direct, maar natuurlijk – niet abrupt. Objecten die worden geactiveerd komen los van hun ondergrond. Hierbij speelt diepte een belangrijke rol. Die diepte wordt bereikt door schaduw op een bepaalde manier toe te passen. Dit is omschreven in een stijlgids van Google: material.io/guidelines/. De eigenschap `box-shadow` is daarbij onmisbaar. Een tweede techniek is transformatie (vervorming) en animatie van blokken. Dat en meer komt aan de orde in het volgende hoofdstuk.



Afbeelding 12.22 Impressie van material design. Het materiaal heeft dikte en bij aanwijzen komt het omhoog (12_05.html).

Bij de afbeelding gaat het in de kern om de volgende code:

```
.material {
  .material > div {
    border-radius: 3px;
    box-shadow: 0 1px 3px 2px rgba(0,0,0,0.12), 0 1px 2px 2px rgba(0,0,0,0.24);
  }
  .material > div:hover {
    box-shadow: 0 10px 30px rgba(0,0,0,0.25), 0 5px 15px rgba(0,0,0,0.22);
    transition: all ease-in-out .2s;
  }
}
```

Afbeeldingen uitsnijden met clip-path

Voor het maken van creatieve uitsneden van (voornamelijk) afbeeldingen is er de eigenschap `clip-path`, afkomstig uit de module *CSS Masking level 1*. `clip-path` vervangt de verouderde eigenschap `clip` en biedt meer mogelijkheden.



Clip is 'uit'

CSS 2.1 bevat de eigenschap `clip`, maar die kunt u beter niet meer gebruiken. Om compatibiliteitsredenen blijft die wel werken, maar `clip` is verouderd.

De term `clip-path` (uitknippad) kent u misschien uit Photoshop of Illustrator. De eigenschap definieert een gebied of een vorm (rechthoek, cirkel, veelhoek) en alles daarbuiten wordt weggeknipt. De volgende code maakt een ruitvormige uitsnede van een foto (afbeelding 12.23):

```
img {
  clip-path: polygon(50% 0, 100% 50%, 50% 100%, 0 50%);
}
```

Met `clip-path` kunnen verschillende gebieden en vormen worden gedefinieerd. Praktisch gebruik is beperkt tot de basis, maar die is voldoende voor gevorderde uitsneden.

- `clip-path: url('afbeelding.svg#id')` een SVG-bronafbeelding, waarvan de vorm dient als het uitknippad. Dat kan een externe SVG zijn of een SVG in de HTML waar met een ID naar wordt verwezen. Het daadwerkelijke uitknippad wordt gedefinieerd door het SVG-element `<clipPath>`.



Afbeelding 12.23 Ruitvormige uitsnede van een foto. Op de achtergrond is de originele foto zichtbaar (12_06.html).

```
<svg>
  <defs>
    <clipPath id="clip-circle">
      <circle cx="310" cy="220" r="200" />
    </clipPath>
  </defs>
</svg>
<div class="clip-url">
  
</div>
.clip-url > img {
  clip-path: url('#clip-circle');
}
```



Afbeelding 12.24 Uitsnede op basis van een SVG-cirkel in de HTML (12_06.html).

- Geometrische basisvormen: `inset()`, `circle()`, `ellipse()`, `polygon()` de vormdefinitie bepaalt de uitsnede en hoeken kunnen worden afgerond. Hier bepaalt dus de eigenschap `clip-path` het uitknippad en niet het element `<clipPath>`.

```
:clip-inset > img {  
    clip-path: inset(0 100px round 20px);  
}
```



Afbeelding 12.25 Een uitsnede met `inset()`, 0 pixels van boven en onder, 100 pixels van links en rechts en met 20 pixels afgeronde hoeken. Het is dezelfde bloem als in afbeelding 12.23 (12_06.html).

In de specificatie *CSS Masking Module level 1* (www.w3.org/TR/css-masking-1/) op vindt u alle mogelijkheden.



Maskeren met mask

Uitsnijden is niet de enige optie voor beeldbewerking (bedenk dat beeld kan ook een blok zonder grafische inhoud zijn, bijvoorbeeld een div met een gekleurde achtergrond). Ook maskeren behoort (in theorie) tot de mogelijkheden. Het probleem is de browserondersteuning. Toegepast op SVG werkt maskeren prima, maar op HTML zeer beperkt. Zie hacks.mozilla.org/2017/06/css-shapes-clipping-and-masking/ voor meer informatie.

Toepassing: header met schuine kanten

Met `clip-path` kunnen ook de rechthoekige boxen een andere vorm krijgen. Een header met een puntvorm kan best leuk zijn. We maken een eenvoudige opzet met HTML:

```
<body>
<header>
<h1>Handboek HTML en CSS</h1>
</header>
<nav>
<ul>
<li>Menu 1</li>
<li>Menu 2</li>
<li>Menu 3</li>
<li>Menu 4</li>
</ul>
</nav>
</body>
```

Het menu is er alleen om te laten zien waar de box van de header eindigt. De complete code is beschikbaar als `12_07.css` en die zal intussen duidelijk zijn, dus nu bekijken we alleen hoe de vorm van de header wordt gemaakt. Daarvoor is de volgende CSS van belang:

```
header {
  clip-path: polygon(50% 100%, 0 35%, 0 0, 100% 0, 100% 35%);
  height: 400px;
}
```



Afbeelding 12.26 Het eindresultaat (12_07.html).

Analyse van de code

Met `clip-path: polygon` wordt een veelhoek getekend met als nulpunt de hoek linksboven van de header, die een hoogte heeft van 400px. Het beginpunt is de punt omlaag op 50% 100%. De percentages betekenen 50% naar rechts en 100% omlaag vanuit de hoek linksboven. Het volgende punt is 0 naar rechts en 35% omlaag. Dan met 0 0 naar de hoek linksboven en met 100% 0 naar de hoek rechtsboven. Het laatste punt blijft op 100% naar rechts en gaat 35% omlaag. En dat is alles.

In de voorbeeldcode is een aanwijseffect op de header gezet. Dat werkt alleen op het zichtbare deel van de header. Buiten het kleurverloop is de header daadwerkelijk weggeknippt. De rand van de oorspronkelijke box is er nog wel, want daar staat het menu tegenaan. De positie van het menu verandert niet als de vorm van de header verandert.

Firefox heeft een hulpmiddel voor het maken van vormen. Selecteer het HTML-element met een geldige declaratie van `clip-path`. In de CSS-weergave staat een pictogram. Klik daarop en er verschijnen versleepbare punten in de paginaweergave. Daarmee kan de vorm worden aangepast. Het resultaat kan worden gekopieerd naar het HTML-bestand.



```
V Stylen filteren
 :hover       :active       :focus       :focus-within
header {
background-image: linear-gradient(to bottom, ● hsl(205, 100%, 50%), ● hsl(90, 100%, 50%));
clip-path: polygon(50% 100%, 0 35%, 0 0, 100% 0, 100% 35%);
```

Afbeelding 12.27 Hulpmiddel voor het maken van `clip-path`-vormen in Firefox.

Beeldfilters

CSS bevat filters grafische effecten op afbeeldingen kunnen worden toegepast. De namen zullen u niet onbekend voorkomen, want ze zijn gangbaar in de beeldbewerking. Tussen haakjes staat de verwachte eenheid. Een filter wordt op dezelfde manier toegepast als elke andere eigenschap. Voorbeelden van alle filters staan in het bestand 12_08.html. Bekijk dat bestand om te zien wat de filters doen, want in een zwart-witafbeelding blijft daar niets van over.

- `filter: blur(lengte)`
- `filter: brightness(nummer of percentage)`
- `filter: contrast(nummer of percentage)`
- `filter: drop-shadow(x-verschuiving, y-verschuiving, vervagingsstraal, spreiding, kleur)`
- `filter: grayscale(nummer of percentage)`
- `filter: hue-rotate(hoek)`
- `filter: invert(nummer of percentage)`
- `filter: opacity(nummer of percentage)`
- `filter: sepia(nummer of percentage)`
- `filter: saturate(nummer of percentage)`

Kleuren mengen

In CSS3 zijn er eigenschappen waarmee kleuren worden gemengd. Het zijn twee eigenschappen met een net iets andere werking. Ze gebruiken wel beide dezelfde waarden voor het mengen. Deze waarden zullen u – net als de beeld-filters – ook niet vreemd voorkomen als u weleens met Photoshop of Gimp hebt gewerkt; het zijn de gangbare (Engelse) termen.

De twee eigenschappen zijn:

- `background-blend-mode` Hiermee worden achtergrondafbeeldingen gemengd met elkaar en de achtergrondkleur van het element.
- `mix-blend-mode` Hiermee wordt de inhoud van het element gemengd met zijn achtergrond en met inhoud van het ouderelement. Het resultaat is dat de kleur van de inhoud verandert.

De waarden zijn:

- `...-blend-mode: normal` (standaard)
- `...-blend-mode: multiply`
- `...-blend-mode: screen`

Isolation

Bij het mengen van kleuren kan het nodig zijn om de achtergrondkleur uit te sluiten. Dan gaat meestal om de achtergrondkleur van het ouderelement (de container). Daarvoor dient de eigenschap `isolation`, die twee waarden heeft:

- `auto` (standaardwaarde)
- `isolate`

Met de waarde `isolate` maakt het element een nieuwe stapelcontext (*stacking context*). Een stapelcontext verwijst naar hoe de elementen op een webpagina op elkaar liggen (denk ook aan `z-index`, waarmee de stapelvolgorde wordt veranderd). Door een nieuwe stapelcontext te maken, tellen onderliggende elementen niet meer mee bij het mengen van kleuren. In de laatste oefening van dit hoofdstuk maakt u drie overlappende gekleurde cirkels, waarbij het belangrijk is dat alleen de kleuren van de cirkels worden gemengd. Met `isolation: isolate` op de container werkt dat perfect.

Samenvatting

In dit hoofdstuk zijn de eigenschappen voor in het oog springende kenmerken besproken: kleur, kleurverlopen, achtergrondafbeeldingen, randen met afbeeldingen en ronde hoeken en schaduw op blokken.

- Voor kleur zijn in CSS verschillende systemen beschikbaar:
 - benoemde kleuren
 - RGB-kleuren
 - HSL-kleuren
 - RGB en HSL met transparantie
- Een element kan doorzichtig zijn, dan schijnt het onderliggende element erin door. De eigenschap heet `opacity`.
- De achtergrond van een element kan een kleur krijgen, maar ook een achtergrondafbeelding: `background-image`. De positie en de zichtbaarheid daarvan zijn volledig instelbaar.
- Een achtergrondafbeelding is niet per se een plaatje, het kan ook een kleurverloop zijn.
- Kleurverlopen zijn er in twee vormen: lineair (`linear-gradient`) en radiaal (`radial-gradient`).
- Randen van afbeeldingen kunnen worden afgerond, symmetrisch maar ook verschillend. Hiervoor is er de eigenschap `border-radius`.
- De rand van een element kan worden gevuld met een afbeelding of kleurverloop. De eigenschap heet `border-image`. De plaatsing van de afbeelding is volledig te beïnvloeden.

- Elk element kan worden voorzien van schaduw met de eigenschap box-shadow. Met subtiele, goed geplaatste schaduwen kan het platte vlak tot leven komen.
- Afbeeldingen kunnen worden bewerkt met behulp van CSS-filters zoals blur() of grayscale().
- Met mengmodi worden (achtergrond)kleuren van element gemengd. Daarvoor bestaan twee eigenschappen die dezelfde waarden gebruiken, maar een ander effect hebben.

De basis van wat u nodig hebt om van kale HTML-elementen een fraai geheel te maken, is nu aan de orde geweest. De manier waarop u elementen selecteert is uitgelegd en u begrijpt nu hoe al de blokken aan hun gedrag komen en hoe u dat kunt manipuleren.

Het volgende hoofdstuk gaat in op geavanceerdere technieken om elementen te bewerken, zoals animatie, transformatie en (kleur)filters. Daarnaast is er de functie calc() die eigenschapswaarden kan berekenen en kunt u CSS-variabelen gebruiken om waarden op te slaan en opnieuw te gebruiken.

Oefeningen



Gebruik de voorbeeldcode

Van vrijwel alle afbeeldingen is de HTML- en CSS-code beschikbaar. De bestandsnaam staat in het bijschrift. Gebruik deze code om te oefenen: pas de code aan, voeg dingen toe, speel ermee! Kijk op handboek-html-css.nl.

- Maak een pagina met een aantal boxen van 200 bij 200 pixels en geef die een achtergrondkleur. Gebruik afwisselend de kleurmodellen RGB en HSL. Het doel hiervan is om te achterhalen welk systeem u het prettigst vindt werken.
- Maak een pagina met een flinke header (een derde van de schermhoogte) en geef die een verlopende achtergrondkleur. Experimenteer met kleuren en met de richting van het verloop.
- Deze oefeningen combineert van alles wat. Het eindresultaat bestaat uit drie elkaar deels overlappend cirkels met de kleuren rood, groen en blauw. Waar ze elkaar overlappen worden de kleuren gemengd.
 - Maak een container-div met daarin drie andere div's.
 - Geef de container een breedte en hoogte, bijvoorbeeld 500 bij 500 pixels.
 - Maak cirkels van de andere drie div's. Daarvoor moeten ze vierkant zijn, bijvoorbeeld 200 bij 200 pixels, en afgeronde hoeken hebben.



Afbeelding 12.29 Drie overlappende cirkels met gemengde kleuren
(12_09.html).

- Geef elke cirkel als achtergrondkleur rood, groen of blauw.
- De cirkels worden absoluut gepositioneerd in de container-div en moeten elkaar deels overlappen. Gebruik daarbij de eigenschappen top en left. (In hoofdstuk 10 staat wat er nodig is om een box absoluut te positioneren in een bepaalde container.)
- Als de gekleurde cirkels op hun plek staan, krijgt elke cirkel de eigenschap mix-blend-mode. Deze eigenschap wordt in het hoofdstuk 13 besproken. Zoek uit welke waarde u nodig hebt om op de overlappende delen de kleuren geel, lichtblauw en paars te zien.
- Tip: geef de container de eigenschap isolation met de waarde isolate.

Overgangen en animaties, calc(), attr() en CSS-variabelen

U hebt inmiddels al veel geleerd over CSS, maar er is altijd meer. In dit hoofdstuk kijken we eerst naar manieren om beweging in uw pagina's te krijgen met overgangen, animatie en transformatie (vervorming). Dan zijn er nog zaken die het werken met de code kunnen vereenvoudigen, zoals de functies calc() en attr(), en CSS-variabelen.

U leert in dit hoofdstuk:

Overgangen maken met de eigenschap transition.

Hoe vloeiente animaties worden gemaakt.

Elementen vervormen met transform, tweedimensionaal en driedimensionaal.

Hoe u het lastige rekenwerk kunt overlaten aan CSS.

Wat CSS-variabelen (CSS custom properties) zijn en hoe ze worden toegepast.

HTML-attributen uitlezen met attr() en gebruiken in de CSS.

Beweging in webpagina's

In modern webdesign speelt beweging en interactie tussen de gebruiker en elementen op het scherm een belangrijke rol. De gebruiker moet zien en bijna voelen dat zijn handelingen iets in gang zetten. Een knop beweegt bij aanraken, een menu schuift in beeld, een afbeelding wordt vergroot enzovoort. Sommige bewegingen gaan in één keer, dan is er een rechtlijnige overgang van de begintoestand naar de eindtoestand, bijvoorbeeld van dicht naar open. Andere bewegingen bestaan uit meerdere stappen, bijvoorbeeld een stuitende knop. Een derde beweging kan bestaan uit het vervormen of draaien van een element.

Voor elk van deze vormen van beweging is er een CSS-eigenschap:

- `transition` in één beweging een overgang van begin naar eind;
- `animation` een beweging in meerdere stappen;
- `transform` vervormen of draaien.

Overgangen (transitions)

Met `transition` maakt u een overgang. Dat kan een veranderende achtergrondkleur zijn, een schaduw, een rand of het formaat van een box of tekst. Ook een combinatie van eigenschappen is mogelijk. De enige beperking is dat de eigenschap *animatable* moet zijn. Dat houdt in dat de waarde moet zijn te herleiden tot een getal. Bij `transition` worden namelijk op basis van de begin- en eindwaarde de tussenliggende waarden berekend (geïnterpoleerd). Dat valt niet mee als de beginwaarde `Times New Roman` en de eindwaarde `Helvetica` is. `font-family` is dus niet *animatable*. De waarde verandert overigens wel als u die opneemt in een overgang, maar niet vloeidend.

Bij een overgang bepaalt u zelf welke eigenschappen worden geanimeerd, wanneer de overgang begint (direct of met vertraging), hoelang de overgang duurt en wat voor soort beweging er is. De verkorte notatie `transition` bundelt alle kenmerken en meestal is het wel zo overzichtelijk om deze te gebruiken. De afzonderlijke eigenschappen zijn (in de gewenste volgorde):

- `transition-property` de eigenschap of eigenschappen voor de overgang;
- `transition-duration` de tijdsduur van de overgang in (milli)seconden;
- `transition-timing-function` het type overgang, zie hierna;
- `transition-delay` de vertragingstijd voordat de overgang begint in (milli)seconden.

Op menu's en knoppen worden vaak overgangen gebruikt. Een aanwijseffect doet het natuurlijk ook zonder transition, maar niet zo vloeiend. Maak iets als het volgende menu en neem de CSS-regels over (code is ook beschikbaar in 13_01.html):

```
<nav class="transition">
  <ul>
    <li><a href="">Home</a></li>
    <li><a href="">Nieuws</a></li>
    <li><a href="">Producten</a></li>
    <li><a href="">Over ons</a></li>
    <li><a href="">Contact</a></li>
  </ul>
</nav>

.transition a {
  display: block;
  padding: .5em;
  font: 700 1.2em sans-serif;
  color: rgb(50,100,150);
  transition: all .5s ease-in-out .05s;
}
.transition a:hover {
  background-color: rgb(50,100,150);
  color: white;
}
```

Bekijk de stijlregels en bekijk de overgang:

```
transition: all .5s ease-in-out .05s;
```

Hier staat: maak een overgang met alle eigenschappen in deze declaratie, die 0,5 seconden duurt, met het effect ease-in-out en een startvertraging van 0,05 seconden. Het sleutelwoord `all` past de overgang toe op alle eigenschappen. Het is geen probleem dat de `:hover`-declaratie veel minder eigenschappen bevat. Eigenschappen die geen nieuwe waarde krijgen, veranderen niet.

Hoe wordt de overgang geactiveerd?

Daar zorgt in dit voorbeeld de `:hover`-klasse voor. Er wordt gewoon een aanwijseffect geactiveerd. Omdat in de hoofdklasse met `transition` is aangegeven dat alle waarden met een overgang mogen worden veranderd, worden de waarden die anders zijn in de `:hover`-klasse niet abrupt maar soepel veran-

derd. Activeren kan ook met klikken op een knop, maar dan is JavaScript nodig.
(De code is beschikbaar in 13_01.html.)

```
<button onclick="classList.toggle('clicked')>Klik op mij</button>
button {
    background-color: tomato;
    color: white;
    font: 700 3em sans-serif;
    padding: 1em;
    transition-property: background-color, color;
    transition-timing-function: ease;
    transition-duration: .5s;
}
.clicked {
    background-color: turquoise;
    color: darkblue;
}
```

De gebeurtenis (event) klikken op de knop (onclick) schakelt beurtelings de klasse .clicked in of uit.

Voor een overgang kunnen in plaats van all ook alleen de betrokken eigenschappen worden genoemd:

```
transition: color .5s ease-in-out .05s, background-color .5s ease-in-out .05s
```

De declaratie is nu wel langer, maar u ziet het vast het voordeel van deze aanpak: de eigenschappen kunnen verschillende overgangen krijgen:

```
transition: color 1s ease-in-out, background-color .5s ease-in-out .05s
```

Door ook op de :hover-klasse een overgang in te stellen, krijgt het aanwijzen een andere overgang dan het weghalen van de muisaanwijzer:

```
.transition a {
    display: block;
    padding: .5em;
    font: 700 1.2em sans-serif;
    color: rgb(50,100,150);
    /*transition: all .5s ease-in-out .05s;*/
    transition: color 1s ease-in-out .05s, background-color 1s ease-in-out .05s
}
.transition a:hover,
.transition a:focus {
```

```
background-color: rgb(50,100,150);
color: white;
transition: all .2s ease-in-out .05s;
}
```

Nu duurt het vervagen van de aanwijskleur veel langer dan het opzetten ervan.

Kant-en-klare overgangen

Voor het type overgang (`transition-timing-function`) zijn sleutelwoorden beschikbaar met vooraf ingestelde kenmerken. Uit de woorden kunt u afleiden wat het effect zal zijn.

- `ease` (standaard)
- `ease-in`
- `ease-out`
- `ease-in-out`
- `linear`
- `step-start`
- `step-end`

Verder kunt u met `steps(getal, start of end)` aangeven in hoeveel stappen de eindtoestand moet worden bereikt, waarbij de tijdsduur en het aantal stappen bepalen hoe vloeiend de overgang verloopt.

De hiervoor genoemde sleutelwoorden hebben overgangen die zijn opgebouwd met een **bézierkromme** (een wiskundige functie). Zo is `ease-in-out` als **bézierfunctie**:

```
transition: all .5s cubic-bezier(0.42, 0.0, 0.58, 1.0) .05s;
```

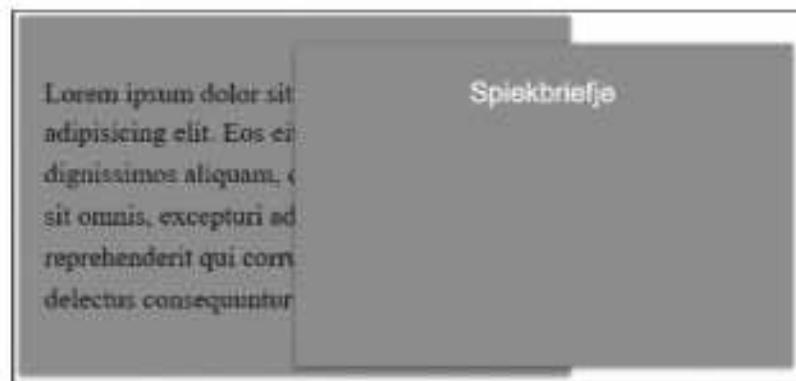
Bézierkrommen zijn een interessant onderwerp, maar vallen zonder twijfel in de categorie geavanceerd. Een verhelderende uitleg van `transition-timing-function` en bézierkrommen is te vinden op www.smashingmagazine.com/2014/04/understanding-css-timing-functions/. Ook het artikel `Timing functions` op developer.mozilla.org/en-US/docs/Web/CSS/single-transition-timing-function is een aanrader.

Positie aanpassen

Tot slot nog een overgang waarbij een blok opzij schuift. Er liggen twee blokken op elkaar en de tekst in het onderste blok wordt afgedekt door het bovenste blok. Door het blok aan te wijzen met de muis schuift het opzij zodat u kunt spieken ([13_02.html](#)).

```
<div class="container">
  <p>Lorem ipsum ... </p>
  <div class="slider">
    <p>Spiekbriefje</p>
  </div>
</div>
```

```
.container {
  width: 400px;
  padding: 1em;
  position: relative;
}
.slider {
  position: absolute;
  top: 1em;
  left: 1em;
  width: 90%;
  height: 90%;
  transition: all .5s ease;
}
.container:hover .slider {
  left: 400px;
}
```



Afbeelding 13.1 Halverwege de verschuiving (13_02.html)

Hier wordt de positie `left` van `.slider` met een overgang veranderd van 1em naar 400px. Met `transition` werkt dit prima, maar verplaatsen kan ook anders: dan gebruikt u een transformatie (zie verderop in dit hoofdstuk).

Animatie

Wanneer voor een overgang meer stappen nodig zijn dan alleen het begin- en eindpunt wordt het tijd om animaties in te zetten. Animaties zijn net als overgangen en transformaties nog een experimentele technologie (de specificatie is in conceptfase), maar moderne browsers kunnen er prima mee uit de voeten. Het is al fantastisch dat met alleen HTML en CSS – zonder scripting – overgangen mogelijk zijn en animaties zijn daarvan de overtuigende trap.

Opbouw een animatie

Voor een animatie hebt u twee dingen nodig:

- het te animeren element met een definitie van de animatie;
- keyframes met de te animeren eigenschappen en de momenten waarop dat moet gebeuren.



Niet alle eigenschappen zijn geschikt

Net als bij overgangen zijn ook niet alle eigenschappen geschikt voor animatie. Met gezond verstand komt u een heel eind, maar bij twijfel kunt u in de specificatie van de CSS-eigenschap terugvinden of die *animatable* is.

We maken een box waarvan de achtergrondkleur met een animatie honderd tinten groen doorloopt (13_03.html).

```
<div class="animate"></div>

.animate {
    animation-name: kleur;
    animation-duration: 2s;
    animation-timing-function: linear;
    animation-iteration-count: infinite;
    animation-delay: initial;
    animation-direction: alternate;
    animation-fill-mode: initial;
    animation-play-state: initial;
}

@keyframes kleur {
    0% {
        background-color: hsl(150, 50%, 100%);
    }
}
```

Animatie

Wanneer voor een overgang meer stappen nodig zijn dan alleen het begin- en eindpunt wordt het tijd om animaties in te zetten. Animaties zijn net als overgangen en transformaties nog een experimentele technologie (de specificatie is in conceptfase), maar moderne browsers kunnen er prima mee uit de voeten. Het is al fantastisch dat met alleen HTML en CSS – zonder scripting – overgangen mogelijk zijn en animaties zijn daarvan de overtuigende trap.

Opbouw een animatie

Voor een animatie hebt u twee dingen nodig:

- het te animeren element met een definitie van de animatie;
- keyframes met de te animeren eigenschappen en de momenten waarop dat moet gebeuren.



Niet alle eigenschappen zijn geschikt

Net als bij overgangen zijn ook niet alle eigenschappen geschikt voor animatie. Met gezond verstand komt u een heel eind, maar bij twijfel kunt u in de specificatie van de CSS-eigenschap terugvinden of die *animatable* is.

We maken een box waarvan de achtergrondkleur met een animatie honderd tinten groen doorloopt (13_03.html).

```
<div class="animate"></div>

.animate {
    animation-name: kleur;
    animation-duration: 2s;
    animation-timing-function: linear;
    animation-iteration-count: infinite;
    animation-delay: initial;
    animation-direction: alternate;
    animation-fill-mode: initial;
    animation-play-state: initial;
}

@keyframes kleur {
    0% {
        background-color: hsl(150, 50%, 100%);
    }
}
```

```

25% {
  background-color: hsl(150, 50%, 75%);
}
50% {
  background-color: hsl(150, 50%, 50%);
}
75% {
  background-color: hsl(150, 50%, 25%);
}
100% {
  background-color: hsl(150, 50%, 0%);
}

```

De klasse `.animate` bevat de definitie van de animatie. Niet alle eigenschappen hebben een ingestelde waarde, maar dit zijn ze allemaal.

- `animation-name` Dit is de naam die dit element koppelt aan `@keyframes`.
- `animation-duration` De totale tijdsduur van de animatie, dat is de tijd waarin alle stappen in `@keyframes` eenmaal zijn doorlopen.
- `animation-timing-function` Dit zijn dezelfde sleutelwoorden als bij `transition: linear, ease, ease-in, ease-out, ease-in-out, cubic-bezier(), steps en frames()`.
- `animation-iteration-count` Bepaalt hoe vaak de animatie wordt afgespeeld. De waarde is een getal of het sleutelwoord `infinite` (oneindig afspelen).
- `animation-delay` Vertragingstijd voordat de animatie begint nadat het startsein is gegeven.
- `animation-direction` De volgorde waarin de animatie wordt afgespeeld:
 - `normal` (de ingestelde volgorde)
 - `reverse` (omgekeerde volgorde)
 - `alternate` (van begin tot eind en terug)
 - `alternate-reverse` (in omgekeerde volgorde van begin tot eind en terug).
- `animation-fill-mode` Standaard worden de eigenschappen in de keyframes buiten de animatietylde niet toegepast op het element (instelling `none`). Dat kan worden aangepast met de volgende waarden:
 - `forwards` het resultaat van het laatste frame blijft zichtbaar;
 - `backwards` het resultaat van het eerste laatste frame blijft zichtbaar;
 - `both` het resultaat van het eerste en het laatste frame blijft zichtbaar.
 Wat het eerste of laatste frame is, hangt af van de richting van de animatie. In ons voorbeeld is het eerste frame wit en het laatste frame zwart.
- `animation-play-state` De animatie loopt (`running`) of is gepauzeerd (`pause`). Zonder scripting kunt u met deze eigenschap weinig beginnen.

Kijken we naar onze kleuranimatie, dan heeft die de naam kleur, is de speelduur 2 seconden, verlopen de overgangen tussen de keyframes lineair, speelt de animatie eindeloos af en doet hij dat eindeloos (*infinite*). Het is niet nodig om alle eigenschap te gebruiken en u kunt het kort houden met de verkorte eigenschap `animation`. De volgorde van de waarden is zoals hier beschreven. Dan wordt de declaratie uit het voorbeeld:

```
.animate {
    animation: kleur 2s linear infinite alternate;
}
```



Meerdere animaties instellen

Het is niet nodig om alle animaties in één keyframedefinitie te zetten. Als u verschillende tijden en animatiemethodes wilt gebruiken kan dat niet eens. In dat geval plaatst u meerdere definities achter `animation`, gescheiden door een komma: `animation: kleur 2s linear, beweging 3s ease-in-out, transform 3s;`

Keyframes

Wat er daadwerkelijk aan beweging te zien is, staat in de stijlregel `@keyframes` (dit wordt een at-rule genoemd, net als `@import` en `@media`). Een keyframedefinitie kan heel uitgebreid zijn als er veel stops zijn waarin veel eigenschappen worden geanimeerd, maar de basis is eenvoudig:

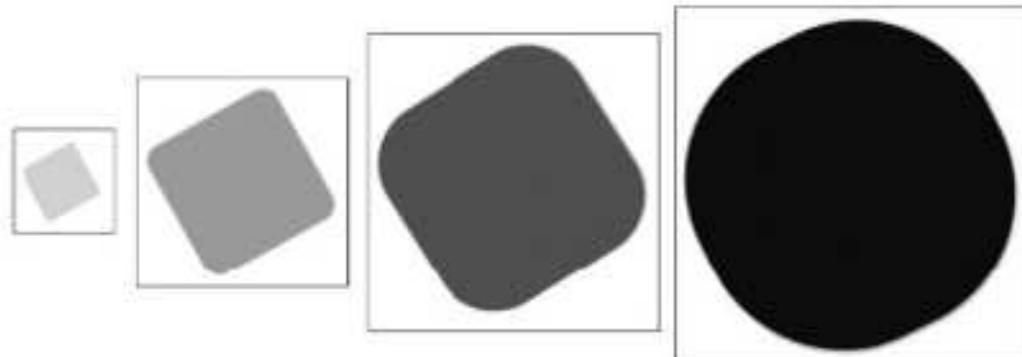
- De regel begint met `@keyframes` gevolgd door de naam van de animatie en een accoladepaar. De naam hebt u zelf bedacht bij `animation-name`.
- Binnen de accolades komen de stops. Er zijn twee sleutelwoorden, `from` en `to`, en er kunnen percentages worden gebruikt. Achter de stops komt ook een accoladepaar.
- Binnen de accolades van de stops komen de CSS-eigenschappen die moeten worden geanimeerd.

In ons voorbeeld staan alleen percentages. Het is niet verplicht om `from` en `to` te gebruiken; ze komen overeen met 0% en 100%. Met alleen `from` en `to` maakt u in feite een transition, een overgang in één stap van begin tot eind.

Het voorbeeld animeert nu maar één eigenschap: de achtergrondkleur. We kunnen dat combineren met bijvoorbeeld twee transformaties (zie de volgende paragraaf) en ronde hoeken.

```
@keyframes kleur {
    0% {
```

```
background-color: hsl(150, 50%, 100%);  
transform: scale(0) rotate(0);  
}  
25% {  
background-color: hsl(150, 50%, 75%);  
border-radius: 0;  
}  
50% {  
background-color: hsl(150, 50%, 50%);  
}  
75% {  
background-color: hsl(150, 50%, 25%);  
}  
100% {  
background-color: hsl(150, 50%, 0%);  
transform: scale(1) rotate(360deg);  
border-radius: 50%;  
}  
}
```



Afbeelding 13.2 Het resultaat: een vorm die verandert van klein naar groot, van vierkant naar cirkel, van wit naar zwart via tinten groen, en weer terug (13_03.html).



Inspiratie

Op daneden.github.io/animate.css staat een uitgebreide verzameling animaties. U kunt ze zo gebruiken in uw projecten, maar het is vooral leerzaam om ze te analyseren. Verbluffende voorbeelden zijn threejs.org/examples/css3d_periodictable.html en codepen.io/juliangarnier/pen/idhuG.

Transformatie

Elementen kunnen worden verplaatst, geschaald, gedraaid en scheefgetrokken en dat kan allemaal ook nog naar keus in het platte vlak of driedimensionaal. Nu zijn zowel de 2D- als de 3D-transformaties nog experimentele eigenschappen waarvan de specificaties nog een concept zijn, maar alle browsers ondersteunen de eigenschappen.

Bij een tweedimensionale transformatie van een HTML-element gaat het om twee eigenschappen:

- `transform` wat gebeurt er met het element;
- `transform-origin` vanuit welk punt in het element gebeurt dat (oorsprong).

Bij een driedimensionale transformatie komt daar nog bij `transform-style`, waarmee een 3D-ruimte wordt gedefinieerd.

We beperken ons nu tot tweedimensionale transformaties en dan kan de opbouw van een declaratie heel eenvoudig zijn. We nemen het spiekbriefje uit de vorige paragraaf erbij en passen dezelfde verschuiving toe, maar nu als transformatie. Alle andere CSS-code blijft hetzelfde (13_03.html).

```
.slider:hover {  
    transform: translateX(400px);  
}
```

Probeer dit uit en u zult zien dat het effect (bijna) hetzelfde is, want `translateX(400px)` betekent: verplaats de `transform-origin` 400px over de x-as (horizontaal). Standaard ligt de oorsprong op het middelpunt van het element, gerekend vanuit buitenrand. (De vorige verplaatsing ging vanuit de hoek linksboven van de container, vanwege de absolute positionering, vandaar het kleine verschil.)

De oorsprong van de verplaatsing kan worden aangepast door `transform-origin` een andere waarde te geven met een lengte, een percentage of de sleutelwoorden `left`, `center`, `right`, `top` of `bottom`. Eén waarde geldt voor x, waarbij de tweede waarde center is. Met twee waarden stelt u x en y afzonderlijk in (in die volgorde). Het effect van een transformatie kan met een andere oorsprong flink veranderen. Denk alleen al aan het verschil tussen draaien van een object rond het middelpunt of een van de hoekpunten.

Positieve waarden gaan van links naar rechts of van boven naar beneden. Met negatieve waarden wordt de richting omgekeerd.



Transform als transition

Uit dit voorbeeld blijkt dat een transformatie prima kan worden gebruikt in een geanimeerde overgang. Dat is ook een veelgebruikte toepassing, maar statische transformaties kunnen ook effectieve blikvangers zijn.

Tweedimensionale transformaties

Wat er precies met een element gebeurt, wordt bepaald door de waarde van de eigenschap `transform`. Dit zijn de mogelijkheden in het platte vlak, dus met alleen een x- en een y-as:

Verplaatsen

- `transform: translate(x-waarde, y-waarde)`
- `transform: translateX(x-waarde)`
- `transform: translateY(y-waarde)`

Verplaatsen kan met lengtewaarden of een percentage.

Schalen

- `transform: scale(x-waarde, y-waarde)`
- `transform: scaleX(x-waarde)`
- `transform: scaleY(y-waarde)`

De waarde bij schalen is een schaalfactor, dus `scale(2)` maakt x en y twee keer zo groot.

Draaien

- `transform: rotate(hoek)`

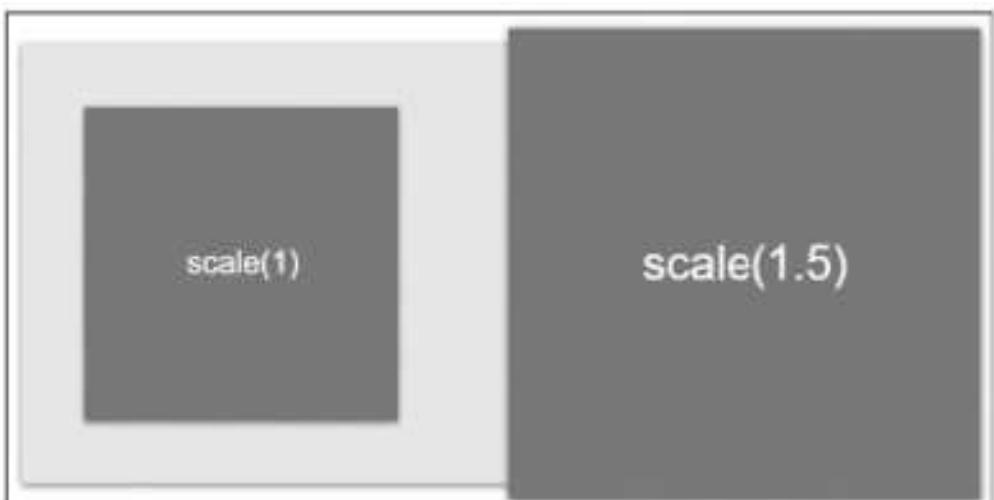
Scheeftrekken

- `transform: skew(x-hoek, y-hoek)`
- `transform: skewX(x-hoek)`
- `transform: skewY(y-hoek)`



Matrix voor de liefhebber

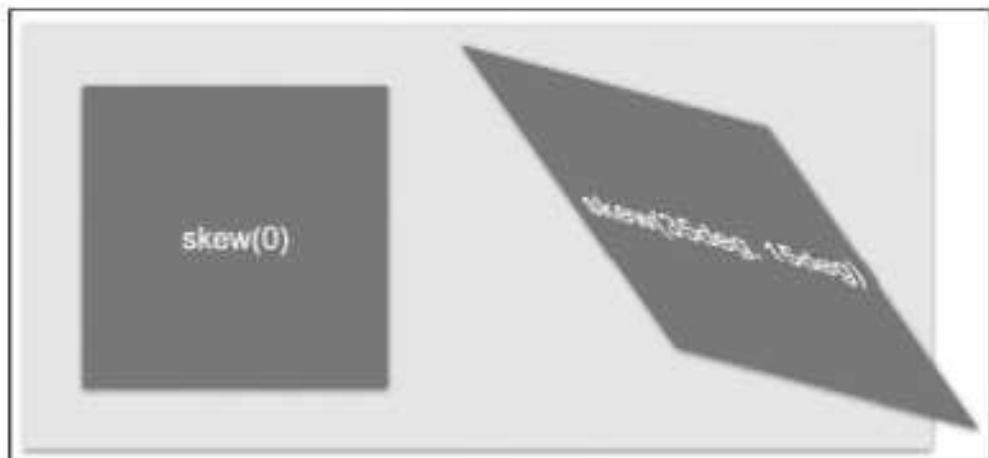
Er ontbreekt een waarde en dat is geen vergissing. Het betreft `matrix()` en die waarde is net zo lastig als `cubic-bezier()` bij transition. De algemene opbouw is `transform: matrix(a, b, c, d, x, y)`. Nu wordt elke transformatie door de browser omgezet in een *transformatiematrix* en die kunt u bekijken in de Hulpmiddelen voor ontwikkelaars. Zo kunt u analyseren hoe het werkt. Een handreiking: a en d zijn voor schalen, b en c voor scheeftrekken en x en y voor verplaatsing. Het voorbeeld `transform: translateX(400px)` geeft `matrix(1, 0, 0, 1, 400, 0)`.



Afbeelding 13.3 Schalen met 1,5. De geschaalde vorm loopt door buiten de container zonder de flow te doorbreken.



Afbeelding 13.4 Een kwart cirkel draaien (0.25turn). Andere eenheden voor hoeken zijn deg en rad.



Afbeelding 13.5 Scheeftrekken met 35deg op de x-as en 15deg op de y-as.

Schalend menu

Met behulp van schalen kan een menu worden gemaakt dat lijkt open te vouwen vanuit de knop waarop wordt geklikt. We maken een eenvoudig menu met een selectievakje. Van dat vakje is alleen het label zichtbaar en aanklikbaar om het menu te openen en sluiten. Deze methode is een alternatief voor JavaScript. Door te klikken op het label wordt het selectievakje ingeschakeld. Met de status :checked kan een selector worden gemaakt, net zoals met de status :hover. Daarmee wordt in dit voorbeeld de menulijst getoond en verborgen.

```
<nav>
  <label for="menu">Menu</label>
  <input id="menu" type="checkbox">
  <ul>
    <li><a href="">Home</a></li>
    <li><a href="">Nieuws</a></li>
    <li><a href="">Producten</a></li>
    <li><a href="">Over ons</a></li>
    <li><a href="">Contact</a></li>
  </ul>
</nav>
```

De weergave van de CSS wordt hier beperkt tot de kern, maar de volledige code is beschikbaar in 13_03.html.

De lijst die het menu bevat neemt natuurlijk ruimte in en zou de tekst eronder omlaag duwen (afbeelding 13.6). Dat is niet de bedoeling en daarom is de <nav> relatief gepositioneerd en de absoluut op 64px vande bovenkant van de <nav>. Daardoor komt die precies onder het label te staan. Vervolgens is de in de hoogte geschaald naar 0 en dat maakt de lijst onzichtbaar.

De oorsprong van de transformatie is verschoven, waardoor die begint net onder de tekst Menu.

De hoogte van de menulijst `` wordt met `transform: scaleY(0)` geminaliseerd tot 0 pixels; onzichtbaar dus. De x-oorsprong van de transformatie wordt ingesteld op het midden, de y-oorsprong komt met `-40px` uit net onder het menulabel. Tot slot wordt een overgang ingesteld voor transform.

```
ul {
    position: absolute;
    top: 64px;
    transform-origin: center -40px;
    transform: scaleY(0);
    transition: all .3s ease-out;
}
```

Nu komt de truc met klikken op het label, waarmee tegelijk het selectievakje wordt ingeschakeld en dan de status checked krijgt. Met een selector voor `` op hetzelfde niveau als de 'checked input' wordt een transformatie geactiveerd naar de normale hoogte van het menu, `scaleY(1)`. Dezelfde techniek zorgt ervoor dat het plusteken bij een geopend menu verandert in een minteken. (Let op de selector ~ voor elementen op hetzelfde niveau maar niet aangrenzend.)

```
input:checked ~ ul {
    transform: scaleY(1);
}
input:checked ~ label::after {
    content: " ";
}
```



Afbeelding 13.6 Het schalende menu terwijl het wordt geopend (13_03.html).

Op vergelijkbare wijze zijn veel meer en ook subtielere transformaties mogelijk. Het menu kan ook buiten het zichtbare venster staan en met translateX van links naar rechts het beeld in schuiven (slider).

Driedimensionale transformaties

De hiervoor beschreven transformaties kunnen ook driedimensionaal worden gemaakt. Dat is in de uitvoering overigens een stuk lastiger.

De 3D-transformaties verlopen langs drie assen: natuurlijk de horizontale x-as en de verticale y-as, aangevuld met de z-as, en die loopt vanuit de kijker naar (door) het beeld. Bij een positieve waarde van de z-coördinaat beweegt het object naar de kijker toe, bij een negatieve waarde gaat het van hem af.

Eij 3D-transformaties hebt u perspectief nodig. Daarmee worden de kijkafstand en de kijkhoek ingesteld. Zonder perspectief worden driedimensionale objecten wel vervormd, maar ze blijven plat.

```
<div class="container">
  <div class="transform"></div>
</div>

.container {
  width: 300px;
  height: 300px;
}
```



Afbeelding 13.7 Links het element normaal geplaatst. Rechts een rotatie van 60 graden over de y-as zonder perspectief. Het element wordt wel smaller, maar is niet driedimensionaal. Met translateY is het element 10px cm laag gezet (13_08.html).

```
.transform {
  width: 300px;
  height: 300px;
  transform: rotateY(60deg) translateY(10px);
}
```

Perspectief

Perspectief wordt op de container van het object ingesteld met de eigenschap `perspective`. Mogelijke waarden zijn `none` of een lengte. Een perspectief van `1000px` is vergelijkbaar met de normale kijkafstand; het object blijft even groot, maar staat wel in een driedimensionale ruimte. Met een kleinere waarde komt de kijker dichterbij en zullen objecten groter lijken. Bij waarden groter dan `1000px` gebeurt het omgekeerde. In dit voorbeeld met rotatie neemt met de kijkafstand de sterkte van het draaieffect af; bij een kleiner perspectief wordt het juist sterker. Bij draaien in 3D kan het perspectief ook worden ingesteld met een functie bij de transformatie. (Als u beide mogelijkheden instelt wordt het effect evenredig versterkt.)

```
<div class="container">
  <div class="transform"></div>
</div>

.container {
  width: 300px;
  height: 300px;
  perspective: 250px; /* of hier */
}
```

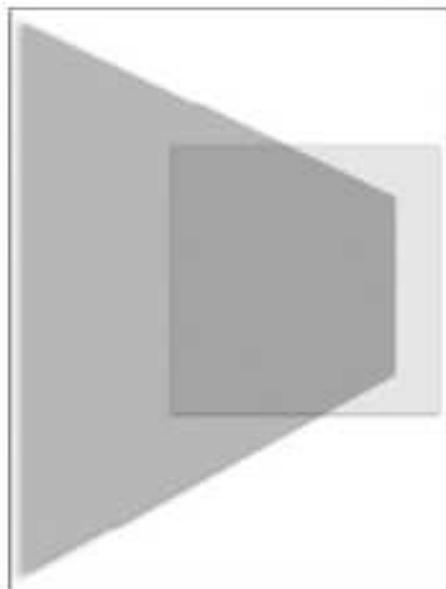


Afbeelding 13.8 Dezelfde rotatie, maar nu met 250px perspectief (13_09.html).

```
.transform {  
    width: 300px;  
    height: 300px;  
    transform: perspective(250px) rotateY(60deg); /* óf hier */  
}
```

Kijkrichting

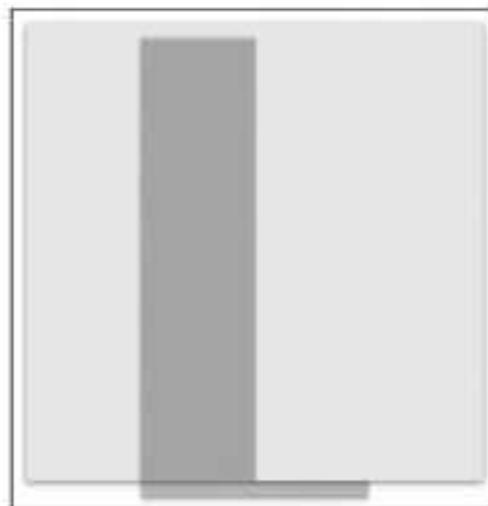
De kijkrichting wordt ingesteld met de eigenschap `perspective-origin` op de container. Standaard is dit het middelpunt op 50% 50%. Mogelijke waarden zijn een percentage, een lengte of de sleutelwoorden `top`, `right`, `bottom`, `left` en `center`. De eerste waarde is voor de x-as, de tweede voor de y-as. Bij één waarde verandert x en blijft y 50%.



Afbeelding 13.9 Opnieuw dezelfde rotatie en hetzelfde perspectief, maar de kijker ziet het vlak van rechts (`perspective-origin: right`). Daardoor lijkt het langer (13_10.html).

3D-ruimte

Dan is er nog de eigenschap `transform-style`. Hiermee wordt ingesteld of de kinderen van een 3D-container in de 3D-ruimte worden geplaatst of in het platte vlak blijven. Met andere woorden: of ze door de container heen mogen steken of er bovenop staan. Er zijn drie waarden: `auto`, `flat` en `preserve-3d`. De standaardwaarde is `auto`, wat neerkomt op `flat`. De instelling `flat` geeft wat u ziet in de afbeeldingen 13.8 en 13.9: het gedraaide element stuit op de container. In de volgende afbeelding is `transform-style: preserve-3d` toegevoegd (en 10px verschuiving op de y-as, omdat het effect dan duidelijker is).



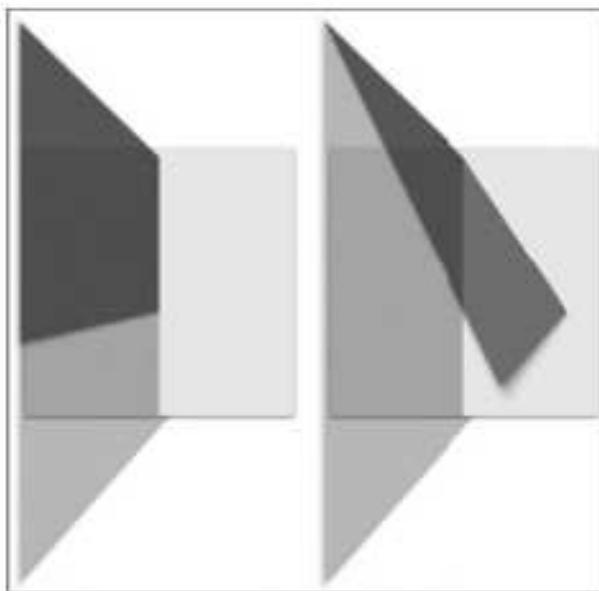
Afbeelding 13.10 Als afbeelding 13.7, maar het grijze vlak heeft zijn eigen 3D-ruimte en steekt door de container (13_11.html).



Afbeelding 13.11 Als afbeelding 13.8, maar net als de afbeelding 13.10 ook met transform-style: preserve-3d (13_12.html).

Deze eigenschap overerft niet, dus kleinkinderen krijgen geen perspectief. Daarvoor moet hun ouder ook de instelling preserve-3d hebben.

```
<div class="container">
  <div class="transform">
    <div class="inner"></div> /* extra kind */
  </div>
</div>
```



Afbeelding 13.12 Links het kleinkind zonder *preserve-3d* en rechts met *preserve-3d* op *div.transform* (13_13.html).

```
.container {  
    transform-style: preserve-3d; /* geldt alleen voor ,transform */  
}  
.transform {  
    transform-style: preserve-3d;  
}
```

Voor driedimensionale transformaties zijn extra transformatie-eigenschappen beschikbaar naast de hiervoor genoemde 2D-eigenschappen:

Verplaatsen

- *transform: translateZ(z-waarde)*
- *transform: translate3d(x-waarde, y-waarde, z-waarde)*

Verplaatsen kan met lengtewaarden of een percentage, maar de z-as kan geen percentage hebben.

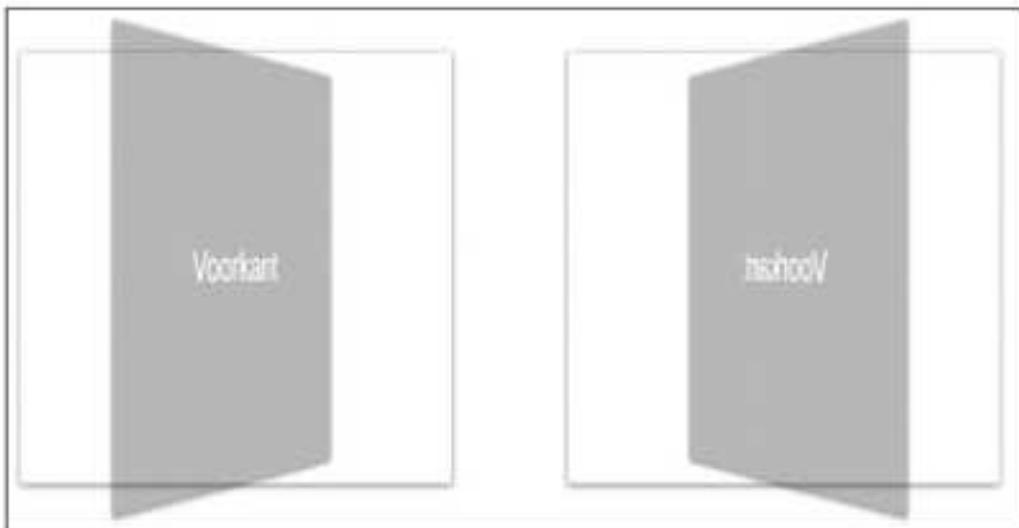
Schalen

- *transform: scaleZ(z-waarde)*
- *transform: scale3d(x-waarde, y-waarde, z-waarde)*

De waarde bij schalen is een schaalfactor.

Draaien

- *transform: rotateZ(hoek)*
- *transform: rotate3d(x-hoek, y-hoek, z-hoek)*



Afbeelding 13.13 Als tijdens de rotatie de achterkant tevoorschijn komt, is de voorkant in spiegelbeeld te zien (13_14.html).

Scheeftrekken

Scheeftrekken over de z-as kan niet.

Zichtbare achterkant

Wanneer objecten in een driedimensionale ruimte bewegen, moeten ze behalve een voorkant ook een achterkant hebben. Nou krijgt een element niet opeens dikte, maar er is wel een soort achterkant. Draai een element langs de y-as en u ziet de voorkant in spiegelbeeld. De achtergrond van de achterkant is doorzichtig en daardoor is de voorkant zichtbaar.

In CSS wordt met de eigenschap `backface-visibility` de zichtbaarheid van die achterkant geregeld. Er zijn maar twee mogelijkheden:

- `backface-visibility: visible` de voorgrond schijnt in spiegelbeeld door in de achterkant;
- `backface-visibility: hidden` de voorgrond is verborgen.

Hoewel er niet een echte achterkant is, kunt u die wel maken. Een eenvoudige methode bestaat uit twee op elkaar gelegde elementen waarvan er een ondersteboven ligt (met `translateY: 180deg` bijvoorbeeld). Van beide elementen is de achterkant verborgen. Op een trigger (`:hover` of `:focus`) wordt de container 180 graden gedraaid, waardoor beide elementen worden omgedraaid. Zo komt het onderste elementen (dat al was gedraaid) met de voorkant boven en verdwijnt het bovenste element uit beeld omdat de onzichtbare achterkant boven komt. In de kern is er niet meer CSS voor nodig dan hier staat. Bekijk het effect op een computer.

```

<div class="bf-wrapper">
  <div class="bf-container">
    <div class="backface front">
      Voorkant
    </div>
    <div class="backface back">
      Achterkant
    </div>
  </div>
</div>

.backface {
  backface-visibility: hidden;
}

.back {
  transform: rotateY(180deg);
}

.bf-wrapper:hover .bf-container {
  transform: rotateY(180deg);
}

```

Rekenen met calc()

CSS3 voegt allerlei handige nieuwe functies toe, waaronder de rekenfunctie `calc()`. Met `calc()` kan worden opgeteld, afgetrokken, vermenigvuldigd en gedeeld. De symbolen zijn `+`, `-`, `*` en `/`. Tussen `+` en `-` en de waarden moet een spatie staan, anders werkt het niet. Het is daarom net zo handig om altijd spaties om de symbolen te zetten. De functie kan als waarde bij een eigenschap worden geplaatst en het resultaat wordt toegepast op de eigenschap.

`calc()` doet niet kinderachtig over de gebruikte eenheden: lengte, frequentie (spraak), hoek, tijd, percentage, nummer en integer zijn allemaal toegestaan. Binnen één type kunnen de eenheden ook door elkaar worden gebruikt. Het is dus mogelijk om in een berekening gebruik te maken van pixels, em, vh en zelfs procent. De enige beperking is dat procent geen speciale betekenis mag hebben, zoals in `background-position`.

Bij het uitvoeren van een berekening gelden de gangbare regels voor de volgorde: eerst vermenigvuldigen en delen, dan optellen en aftrekken, maar het is meestal duidelijker en veiliger om haakjes te gebruiken (zie de voorbeelden).

Het nut van `calc()` wordt nog groter als eigenschappen kunnen worden uitgelezen met de functie `attr()`. Zover is het echter nog niet. Wel kunnen CSS-variabelen worden gebruikt.

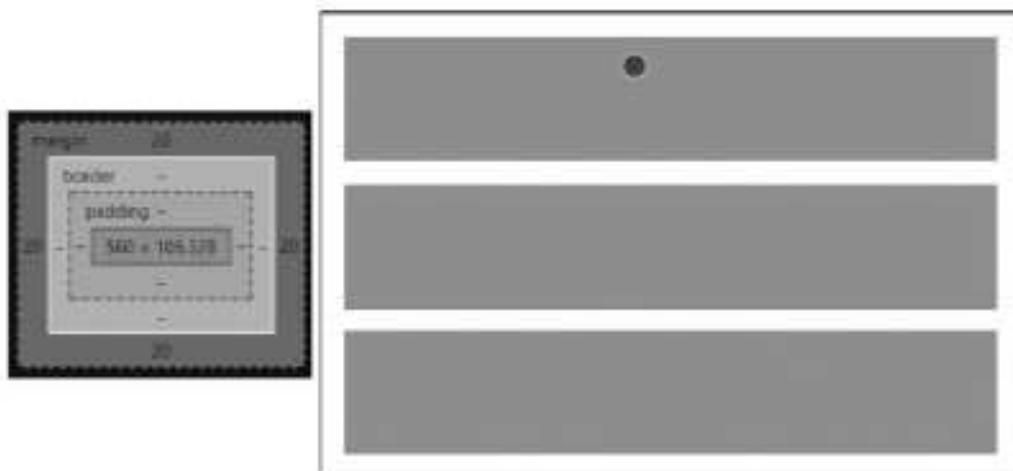
Voorbeelden

Stel dat u drie even hoge blokken wilt hebben met een marge van 1em, die samen precies de viewport vullen. Met een hoogte van 33vh per blok komt u niet uit, want de marge komt er nog bij. Bovendien mist u 1vh! U hebt calc() hier zeker nodig om een oplossing te krijgen die bij elke vensterhoogte werkt, maar hoe?

- De hoogte is bekend, 100vh. Die moet door drie worden gedeeld (100vh / 3).
- Drie blokken met rondom 1em marge geeft vier keer een marge van 1em; aangrenzende marges worden immers ingeklapt (hoofdstuk 9). Die marges moeten over drie blokken worden verdeeld: $((4 * 1em) / 3)$.

De declaratie wordt:

```
div {
    height: calc( (100vh / 3) - ((4 * 1em) / 3) );
}
```



Afbeelding 13.14 Tot op drie cijfers nauwkeurig en altijd passend dankzij calc().

CSS-variabelen: de functie var()

CSS-variabelen zijn een toevoeging aan CSS waar lang naar is uitgekeken. De officiële naam is *CSS Custom Properties*, aangepaste eigenschappen. Het is ook een functie, geschreven als var(). De module heet *CSS Custom Properties for Cascading Variables level 1* (drafts.csswg.org/css-variables/). Hoewel het nog een concept is, is de functie dus al ingevoerd in de browsers.

U kent variabelen misschien uit programmeertalen zoals JavaScript of uit CSS-preprocessors zoals Sass. Een algemene definitie van een variabele is dat het een opslagplaats is voor een waarde waarnaar met een naam kan worden ver-

wezen. In grote lijnen werken CSS-variabelen op dezelfde manier. Het verschil: het is in feite een ‘eigenschap’ waarvoor u zelf een naam mag bedenken en waarin u elke geldige CSS-waarde kwijt kunt. De variabele gebruikt u vervolgens bij een bestaande CSS-eigenschap als waarde. Die moeten dus wel bij elkaar passen. In width kunt u geen kleurwaarde kwijt. De gevolgen zijn overigens beperkt. Er volgt geen crash of syntax error; de declaratie wordt gewoon genegeerd, net als elke ongeldige CSS-declaratie.



De waarde is rekbaar

Een nauwkeuriger uitleg is dat bij de eigenschap waar de variabele wordt gebruikt, de waarde, eventueel in combinatie met andere waarden, een geldige declaratie moet opleveren. Een voorbeeld: we maken een variabele basiskleur met daarin een hsl-kleurwaarde. Die kan later als transparante kleur worden toegepast. De declaratie van de variabele is `--basiskleur: 0,50%,50%;` Toegepast als transparante achtergrondkleur wordt dat `background-color: hsla(var(--basis-kleur),0.5);` Het resultaat wordt `background-color: hsla(0,50%,50%,0.5);`

Een belangrijke toepassing van CSS-variabelen is het vastleggen van terugkerende waarden. Denk aan het kleurenschema van de site; waarden die vaak worden gebruikt en daarmee ook een bron van fouten kunnen zijn. Lettertypen komen ook aanmerking, en natuurlijk wat u zelf handig vindt. De waarden worden één keer vastgelegd in variabelen en in de rest van de CSS gebruikt u die variabele. Moet een kleur of lettertype voor de hele site worden aangepast, dan hoeft alleen de desbetreffende variabele te worden veranderd. Dit is eenvoudiger en het helpt eigenschappen consequent toe te passen.

Kenmerken van variabelen

Een CSS-variabele kan worden veranderd terwijl de pagina al wordt getoond en dan is het resultaat direct zichtbaar. Dat kan met elke eigenschap – een muisovereffect of een media query kan ook waarden veranderen – en zoals gezegd zijn CSS-variabelen in de basis gewoon eigenschappen.

Die basis geldt ook voor het werkingsgebied van variabelen (in een programmeertaal zoals JavaScript heet dat *scope*). CSS-variabelen worden standaard overgeerfd en volgen de cascade (zie hoofdstuk 8). Net als een eigenschap is een variabele gebonden aan een selector. U kunt niet zomaar los in de stylesheet een variabele schrijven die voor elk element moet gelden (JavaScript: globale variabele). U kunt wel een globale CSS-variabele maken, maar dan moet die ook aan een globale selector zijn gekoppeld. Voor HTML is dat de selector `:root` (zie hoofdstuk 8). Dit is in feite het element `<html>`, maar met een hogere specificiteit.

U beperkt CSS-variabelen tot een afgebakend werkgebied (scope) door ze te koppelen aan een specifieke selector, bijvoorbeeld een klasse. De geldigheid van de waarden blijft dan beperkt tot die klasse.

Variabelen declareren

U maakt een variabele door twee streepjes te plaatsen voor een zelfgekozen naam. Een variabele wordt net zo gedeclareerd als elke andere eigenschap:

```
--naam-variabele: waarde;
```

Om de variabele te gebruiken plaatst u de zelfgekozen naam tussen haakjes achter het sleutelwoord var:

```
eigenschap: var(--naam-variabele);
```

De naam van een variabele is hooflettergevoelig: --kleur is een andere variabele dan --Kleur. Het zijn wel beide geldige namen.

Toepassingen van variabelen

Met variabelen kunt u echt alle kanten op. We geven wat voorbeelden van toepassingen.

Globale variabelen

Zoals gezegd is het kleurenschema van een site een voor de hand liggende toepassing. Variabelen die gelden voor de hele site worden gedeclareerd in de selector :root.

```
:root {
  --color-border: hsl(100, 50%, 50%);
  --color-background: hsla(100, 50%, 50%, .5);
  --color-text: #111;
}
```

De toepassing kan er zo uitzien. Deze code resulteert in een <div> met een lichtgroene achtergrond, een donkergrone bovenrand en bijna zwarte tekst.

```
div {
  border-top: 2em solid var(--color-border);
  background-color: var(--color-background);
  color: var(--color-text);
}
```

```
.btn {
  --box-shadow-color: orange;
  box-shadow: 0 0 20px var(--box-shadow-color);
}

.btn:hover {
  --box-shadow-color: blue;
}
```

Variabelen met calc()

Tot slot nog de mogelijkheid om variabelen te gebruiken in berekeningen. Eerder is met calc() de grootte van drie beeldvullende blokken berekend:

```
div {
  height: calc( 100vh / 3 ) - ((4 * 1em) / 3 );
}
```

Stel dat de marge varieert per type blok:

```
.boeken {
  --marge: 2em;
}

.foto {
  --marge: 3em;
}
```

Nu kan de berekening worden gemaakt in de context van de declaratie:

```
div.boeken {
  height: calc( 100vh / 3 ) - ((4 * var(--marge)) / 3 );
}
```

Attributen uitlezen met attr()

De CSS functie attr() maakt het mogelijk om de waarde van een HTML-attribuut te gebruiken in een stylesheet. Het moet ooit bruikbaar zijn voor elk HTML-attribuut en voor elke CSS-eigenschap, maar zo ver is het nog niet. Volgens de specificatie kan attr() allerlei waarden doorgeven, waaronder string, color, url, px, em, %. Dat zou bijvoorbeeld zo kunnen worden toegepast:

```
<div class="attributen" data-background-color="green" data-font-size="50">Lorem ipsum ... </div>

.attributen {
  background-color: red;
  background-color: attr(data-background-color color, yellow);
```

```
font-size: attr(data-font-size px, 50);
}
```

De opbouw van attr() is: eerst een HTML-attribuut, dan een waarde (color, px) en dan een komma en een terugvalwaarde (yellow, 50)

In dit voorbeeld moet de div een groene achtergrondkleur (data-background-color="green") en 50px tekst hebben (data-font-size="50"), maar de browser geeft een rode div met 16px tekst. Al die nieuwe waarden werken nog niet. Een tekenreeks (string) is de enige waarde die wel werkt, en dat is al zo sinds CSS 2.1. Het praktisch nut is daardoor beperkt. Het wordt wel gebruikt op de CSS-eigenschap content in combinatie met de pseudo-klassen ::before en ::after. Het is overigens wel mogelijk om allerlei HTML-attributen uit te lezen, maar de waarde is altijd een tekenreeks, geen kleurwaarde of een nummer.

De volgende code toont de breedte en hoogte van een foto in die foto (probeer het met een eigen foto):

```
<div class="foto-wrapper">
  <div class="foto" data-height="3264px" data-width="2448px">
    
  </div>
</div>
/* CSS */
.foto-wrapper {
  position: relative;
  width: 50%;
}
.foto::before {
  content: "breedte: " attr(data-width) ", hoogte: " attr(data-height);
  font: 20px sans-serif;
  position: absolute;
}
img {
  width: 100%;
  height: auto;
}
```

Het is wel mogelijk om de attributen width en height van uit te lezen, maar bij kunnen ::before en ::after niet worden gebruikt. Daardoor kan de tekst niet als content bij worden geplaatst. Met attr() worden alleen attributen uitgelezen van het element waar de CSS-declaratie bij hoort.

All reset alles, behalve variabelen

Er bestaat een eigenschap waarmee alle eigenschappen van een element in één keer kunnen worden gereset, de eigenschap `all`. Stel dat u een onderdeel of component bouwt en echt helemaal schoon wil beginnen, dan gebruikt u `all: initial`. Daarmee gaan alle eigenschappen terug naar de initiële instellingen volgens de CSS-specificatie. Andere waarden voor `all` zijn `inherit`, `unset` (zie ook hoofdstuk 8) en `revert` (nog niet geïmplementeerd). CSS-variabelen worden niet gereset met `all`.

De toepassing is net als elke andere CSS-eigenschap:

```
button {  
    all: initial;  
}
```

Samenvatting

In dit hoofdstuk zijn handige nieuwigheden van CSS3 besproken en is het geavanceerdere uiterlijk vertoon aan bod gekomen.

- Een overgang in één (vloeiente) stap van een begintoestand naar een eindtoestand wordt gemaakt met de eigenschap `transition`.
- Een overgang met tussenstappen wordt gemaakt met `animate`. Hiermee zijn complexe animaties mogelijk.
- De stappen van een animatie worden vastgelegd in `keyframes`.
- Met transformaties kunnen elementen worden geschaald, gedraaid, verplaatst en scheefgetrokken. Transformaties worden gemaakt met de eigenschap `transform`.
- Transformaties kunnen worden uitgevoerd in het platte vlak (tweedimensionaal) en in de driedimensionale ruimte.
- Met de functie `calc()` kunnen in de stylesheet berekeningen worden gemaakt. Het betreft de basisbewerkingen optellen, aftrekken, delen en vermenigvuldigen. De uitkomsten worden direct toegepast, bijvoorbeeld om bij schalende pagina's tekstgrootte, marges en de grootte van blokken in balans te houden.
- CSS-variabelen (CSS custom properties) maken het mogelijk om veelgebruikte eigenschappen en waarden vast te leggen onder een zelfgekozen naam en deze overal in de stylesheet te gebruiken.
- In CSS-variabelen kunnen ook `calc()` en andere variabelen worden gebruikt.

Oefeningen



Gebruik de voorbeeldcode

Van vrijwel alle afbeeldingen is de HTML- en CSS-code beschikbaar. De bestandsnaam staat in het **bijschrift**. Gebruik deze code om te oefenen: pas de code aan, voeg dingen toe, speel ermee!

Kijk op handboek-html-css.nl.

- Gebruik `calc()` om enkele blokken met marge, padding en border even groot te verdelen in de halve schermhoogte.
- Maak een kleurschema, wijs de waarden toe aan variabelen en pas de kleuren toe aan diverse elementen. Gebruik globale variabelen en variabelen in een scope.
- Maak met transition een animatie voor een menu. Maak een verticaal menu dat vanuit de linkermarge in beeld schuift (kijk voor inspiratie op material.io/design/components/navigation-drawer.html#).
 - Pas de checkbox-methode toe (`:checked`) of koppel de actie aan een mouseover (`:hover`). De checkbox-methode werkt ook op een smartphone.
- Maak een pagina waarop vier blokken worden uitgerekt en geroteerd.
 - Zorg voor rotatie van respectievelijk 15, 45, 120 en 135 graden.
 - Zorg voor scheeftrekken (`skew`) met 120 graden en -120 graden. Wat is het effect van een negatief getal?
 - Geef de blokken duidelijke randen en achtergrondkleuren, zodat ze goed te zien zijn.
- Maak een eigen animatie met keyframes. Begin eenvoudig door bijvoorbeeld de grootte van het lettertype geleidelijk te veranderen.
 - Test met verschillende waarden voor `animation-duration`, `animation-direction` en meer.
- Maak een animatie waarbij een element in een golfbeweging van links naar rechts door het beeld beweegt. Tip: gebruik `transform: translate()`.
 - Als u zich echt wilt laten gaan: laat het element bij elke golfbeweging een kwartslag kantelen.

Index

 **Yendo** Tip: doorzoek de elektronische versie van dit boek kosteloos op yin.do/0818d

:important 223
2D-transformatie
 Zie Transformatie
3D-transformatie 478
 perspectief 479
 perspectiefoorsprong 480
 transformatiestijl 480
 zichtbaarheid achterkant 483
404-pagina 117
::after 249
::before 249
::first-letter 249
::first-line 248
::grammar-error 250
::marker 251
::placeholder 250
::selection 250
::selection-inactive 250
::spelling-error 250
::active 245
::checked 246
::disabled 246
::empty 248
::enabled 246
::first-child 247
::first-of-type 247
::focus 245
::hover 245
::lang() 246
::last-child 247
::last-of-type 247
::link 244
::not(x) 248
::nth-child() 246
::nth-last-child() 246
::nth-last-of-type() 247

::nth-of-type() 247
::only-child 247
::only-of-type 248
::root 246
::target 245
::visited 244
@font-face 389-390
 format (attribuut) 392
@import 232
@keyframes 471
a (element) 97
AAC 139
abbr (element) 74
Absolute (position) 300
accept (attribuut) 178
Accessibility 43
Accessible Rich Internet Applications 42
accesskey (attribuut) 38
Achtergrond
 kleurverloop 440
 meerdere afbeeldingen 439
Achtergrondafbeelding 431
 herhalen 432
 schermpositie 433
 vastzetten 433
Achtergrondkleur 431
action (attribuut) 156
address (element) 76
Afbeelding 84
 afmetingen 128
 bitmaps 119
 GIF 122
 goede alt-tekst schrijven 127
 hoge resolutie 130
 JPEG 123
 klikbare gebieden 129

Index

PNG 124
responsive 125
src 127
vector 120
Afbeelding:WebP 125
Afstammingscombinator 251
align-content (eigenschap) 350
align-items (eigenschap) 348, 369
align-self (eigenschap) 348, 369
Alinea 70
all (eigenschap) 491
alt (attribuut) 127
Animatable 464
Animatie 469
 eigenschappen 470
 keyframes 471
animation (eigenschap) 470
animation-delay(eigenschap) 470
animation-direction (eigenschap) 470
animation-duration (eigenschap) 470
animation-fill-mode (eigenschap) 470
animation-iteration-count (eigenschap) 470
animation-name (eigenschap) 470
animation-play-state (eigenschap) 470
animation-timing-function (eigenschap) 470
API 7, 46
Application Programming Interface 46
Application programming interface 7
area (element) 129
aria-* (attribuut) 42
article (element) 55
ASCII 91
aside (element) 58
attr() (functie) 489
Attribuut 36
 accesskey 38
 boolaans 37
 class 38
 contenteditable 38
 data-* 39
 dir 39
 draggable 39
 globaal 37
 hidden 40
 id 40
 kenmerken 36
 lang 39
 spellcheck 40
 style 40
 tabindex 41
 title 41
 translate 41
Attribuutselector 243
Audio 145
 bestandstypen 145
 audio(element) 145
 auto-fill (sleutelwoord) 345
 auto-fit (sleutelwoord) 345
 autocomplete (attribuut van formulier) 157
 autocomplete (attribuut van invoerveld) 169
 autofocus (attribuut) 159, 164
Automatische nummering 249
backface-visibility (eigenschap) 483
background (eigenschap) 438
background-attachment (eigenschap) 433
background-blend-mode (eigenschap) 458
background-clip (eigenschap) 435
background-color (eigenschap) 431
background-image (eigenschap) 432
background-origin (eigenschap) 436
background-position (eigenschap) 433
background-repeat (eigenschap) 432
background-size (eigenschap) 437
Basisstructuur document 46
BFC 282, 310
Bijschrift 84
Bitmap 119
Bladwijzer 101
 andere pagina zelfde site 101
 binnen een pagina 101
 naar andere site 101
blend-mode (eigenschap) 458
block(display) 282
Block formatting context 282, 310
block-size (eigenschap) 277
blockquote (element) 79
blur() (filter) 458
body (element) 49
border (eigenschap) 269, 273
border-bottom (eigenschap) 272
border-box (box-sizing) 260
border-color (eigenschap) 269
border-image (eigenschap) 447
border-image-outset (eigenschap) 449
border-image-repeat (eigenschap) 449
border-image-slice (eigenschap) 448

border-image-source (eigenschap) 448
 border-image-width (eigenschap) 448
 border-left (eigenschap) 272
 border-radius (eigenschap) 445
 border-right (eigenschap) 272
 border-style (eigenschap) 271
 border-top (eigenschap) 272
 border-width 269
 border-width (eigenschap) 269
 bottom (eigenschap) 299
 Box alignment 279
 box-shadow (eigenschap) 450
 box-sizing (eigenschap) 260
 Boxmodel 259
 hoogte en breedte 260
 br (element) 71
 break-after (eigenschap) 319
 break-before (eigenschap) 319
 break-inside (eigenschap) 319
 brightness() (filter) 458
 Browser 12
 Apple Safari 16
 Chrome Canary 14
 Google Chrome 13
 Internet Explorer 14
 Mozilla Firefox 15
 Browser engine 13
 button (element) 180
 calc() (functie) 484
 Canary 14
 caption (element) 200
 Cascade (begrip) 223
 Cascading Style Sheets
 Zie ook CSS
 Centreren
 horizontaal 265
 verticaal 266
 ch (eenheid) 219
 Chrome 13
 cite (element) 73
 class (attribuut) 38
 Client-side 8
 clip-path (eigenschap) 453
 code (element) 75
 Codec 139
 colgroup (element) 202
 Collapsing margins 267
 color (eigenschap) 430
 colspan (attribuut) 198
 column-count (eigenschap) 312
 column-fill (eigenschap) 318
 column-gap (eigenschap) 316
 column-rule (eigenschap) 317
 column-rule-color (eigenschap) 317
 column-rule-style (eigenschap) 317
 column-rule-width (eigenschap) 317
 column-span (eigenschap) 315
 column-width (eigenschap) 312
 columns (eigenschap) 312
 Combinatieselector 251
 Commentaar (CSS) 231
 Commentaar in HTML 31
 Containing block 297
 content-box (box-sizing) 260
 contenteditable (attribuut) 38
 Contentmodel 52
 categorieën 52
 contents (display) 283
 contrast() (filter) 458
 CSS 3
 boxmodel 259
 cascade 223
 commentaar 231
 declaratie 214
 definitie 10, 210
 eenheden, absoluut en relatief 217
 eigenschap 213
 extern bestand 230
 functies 441
 height 260
 HTML-boomstructuur 221
 normalizer 232
 overerving 226
 reset 232
 selector 213, 234
 specificiteit 224
 stijlregel 213
 waarde 213
 waarden van eigenschappen 215
 weergavemodel 280
 width 260
 CSS custom properties 485
 CSS-bestand 231
 CSS-pixel 217
 CSS-validator 20

Index

CSS-variabele 485
combineren met calc() 489
eigenschappen emuleren 488
globaal 487
kenmerken 486
scope 486
scoped 488
CSS-waarde
getallen 216
globale waarden 217
kleur 216
lengte 216-217
percentage 216
sleutelwoorden 215
URL 216
CSS3
definitie 211
currentColor (kleurwaarde) 270
data-* (attribuut) 39
datalist (element) 184
dd (element) 88
Definities 88
del (element) 76
dense (waarde) 346
Device pixel ratio 131, 297
dfn (element) 89
dir (attribuut) 39
disabled (attribuut) 167
display
grid 330
display (eigenschap) 281, 362
display:flex 362
div (element) 90
dl (element) 88
DNS 112
DOCTYPE (element) 47
Document Object Model 50
DOM 50
dpcm 294
dpi 294
dppx 294
draggable (attribuut) 39
drop-shadow() (filter) 458
dt (element) 88
Edge (browser) 14
Element
attributen 36
Elementselector 240
em (eenheid) 218
em (element) 73
embed (element) 138
Embed code 117
Embedded content 53, 116
enctype (attribuut) 157
Engine 13
Expliekte raster 342
Externe inhoud
HTML-pagina 135
FAQ 38
Favicon 109
fieldset (element) 186
figcaption (element) 84
figure (element) 84
file :// (protocol) 118
Filter voor beeldbewerking 458
Firefcx 15
fit-content() (waarde) 264, 341
fixed(position) 302
flex (display) 283
flex (eigenschap) 363
Flex formatting context 362
flex-basis (eigenschap) 363
flex-direction (eigenschap) 360
flex-flow (eigenschap) 362
flex-grow (eigenschap) 363
flex-shrink (eigenschap) 363
flex-wrap (eigenschap) 361
Flexbox 359
assenstelsel 359
automatische marges 366
hoofdas 359
kenmerken 359
kruisas 359
uitlijnen met automarge 366
verschil met grid-lay-out 326, 359
Flexcontainer 362
Flexitem 362
float (eigenschap) 305
flow (display) 282
Flow content 52
footer 61
header 60
flow-root (display) 282
Font 403
font (eigenschap) 395, 402

Index

- track-list 340
- verschil met flexbox 326, 359
- Grouping content 76
- H.264 139
- h1-h6 (element) 59, 67
- head (element) 48
- header (element) 60
- Heading content 52
- height (attribuut) 128
- height (eigenschap) 260, 263
- hidden (attribuut) 40
- Horizontaal centreren 265
- hr (element) 89
- HSL 428
- HSLA 428
- HTML
 - boomstructuur van de pagina 50
 - broncode in browser bekijken 33
 - commentaar 31
 - definitie 9
 - functie 9
 - geschiedenis 2
 - ontwikkeling 4
- html (element) 47
- HTML 5
 - basisstructuur document 46
 - bitmaps op canvas 45
 - contentmodel 51-52
 - foutafhandeling 44
 - gegevensopslag 46
 - geolocatie 46
 - Zie ook HTML
 - kenmerken 43
 - multimedia 45
 - nieuwe formulierelementen 44
 - nieuwe structuurelementen 43
 - paginastructuur 50
 - scheiding inhoud en opmaak 43
 - service workers 45
- HTML Living Standard 3
- HTML-bestand
 - extensie 28
- HTML-document
 - titel 48
- HTML-editor 16
 - Visual Studio Code 17
- HTML-element
 - attributen 36
- leeg element 30
- opbouw 28, 30
- tag 31
- HTML-validator 20
- https // (protocol) 116
- hue-rotate() (filter) 458
- Hulpmiddelen voor ontwikkelaars 20
- Hyperlink
 - beperkingen 97
 - bestemming 98
 - bladwijzer 101
 - definitie 96
 - download 99
 - href 98
 - hreflang 99
 - lees meer... 97
 - Zie ook Link
 - naar e-mailadres 104
 - naar grote foto 102
 - onderstrekking verwijderen 407
 - openen in nieuw venster 99
 - rel 100
 - rev 100
 - richtlijnen 105
 - target 99
 - title 100
 - type 100
 - vervolg van artikel 103
- Hyperlink:naar telefoonnummer 104
- Hypertekst 9
 - hyperlink 9
- Hypertext Markup Language 2
 - Zie HTML
- hyphens (eigenschap) 413
- id (attribuut) 40
- ID-selector 243
- iframe
 - beveiliging 137
 - src 135
 - srcdoc 136
- iframe (element) 135
- Illustratie 84
- Implicit raster 342
- index.html 99
- Ingesloten stijl 229
- Inheritance 226
- inline(display) 282
- inline-size (eigenschap) 277

- Inner display type 281
- input
 - autocomplete (attribuut) 169
 - autofocus (attribuut) 164
 - disabled (attribuut) 167
 - list (attribuut) 169
 - max (attribuut) 169
 - maxlength (attribuut) 164
 - min (attribuut) 169
 - minlength (attribuut) 164
 - multiple (attribuut) 168
 - name (attribuut) 163
 - pattern (attribuut) 165
 - placeholder (attribuut) 168
 - readonly (attribuut) 167
 - required (attribuut) 168
 - size (attribuut) 167
 - step (attribuut) 169
 - type="button" 180
 - type="date" 175
 - type="file" 178
 - type="number" 172
 - type="radio" 176
 - type="time" 175
 - value (attribuut) 164
- input (element) 162
- ins (element) 76
- Inspringen 407
- Interactive content 53
- Internet Explorer 14
- invert() (filter) 458
- is (attribuut) 42
- isolation (eigenschap) 460
- JavaScript 7
- JPEG 123
 - compressie 123
 - kenmerken 124
- JPG 122
- JSON 7
- justify-content (eigenschap) 350, 367
- justify-items (eigenschap) 348
- justify-self (eigenschap) 348
- kbd (element) 75
- Kindcombinator 252
- Kindselector 252
- Klasseselector 242
- Kleur
 - transparantie 429
- achtergrond 431
- HSL 428
- HSLA 428
- mengen met blend-mode 458
- RGB 427
- RGBA 427
- tekst 430
 - transparantie 428
 - waarden 426
- Kleursysteem 426
- Kleurverloop 441
 - achtergrond 440
 - lineair 441
 - radiaal 443
 - repeterend 444
- Kop 67
 - paginaheader 68
- Koppeling
 - Zie Hyperlink
- Koptekst 59
- label (element) 161
- lang (attribuut) 39
- Lay-out
 - float 307
 - kolommen 312
 - multi-column 312
 - positionering 297
- Leeg element 30
 - leeg ~ 31
- left (eigenschap) 299
- legend (element) 186
- Lengte 217
- letter-spacing (eigenschap) 412
- Lettergrootte
 - eenheden 396
- Lettertype 386, 395
 - @font-face 389
 - downloaden 389
 - embedcode 393
 - formaat 396
 - formaat aanpassen 399
 - gewicht 401
 - kleinkapitaal 404
 - regelhoogte 399
 - stijl 401
 - systeem 395
 - uitrekken of indrukken 403
 - web fonts 387

Index

- li (element) 79
- Lijst
 - cijfers of letter 80
 - geordend 80
 - metadata 88
 - navigatie 416
 - navigatiemenu 83
 - nesten 81
 - ongeordende 82
 - opmaak 414
 - opsommingstekens 415
 - opsommingstekens 82
 - startnummer 80
 - titel 83
 - van afkortingen 89
 - van veelgestelde vragen 88
 - volgorde 80
- line-height (eigenschap) 399
- Lineair kleurverloop 441
- Link
 - alternatieve stylesheet 109
 - favicon 109
 - Zie ook Hyperlink
 - inhoud vooraf laden 111
 - opeenvolgende pagina's 110
 - rel=stylesheet 108
 - stylesheet 108
- link (element) 49, 108, 230
- list (attribuut) 169
- list-item (display) 283
- list-style (eigenschap) 414
- Logische eigenschapper 276
- Logo 84
- Lokale site 118
- mailto: 104
- main (element) 77
- map (element) 129
- Marge 265
 - automatisch 265
 - eigenschappen 265
 - negatief 267
 - samengevoegd 267
- margin (definitie) 259
- margin (eigenschap) 265
- mark (element) 75
- mask (eigenschap) 455
- Material design 452
- max (attribuut) 169
- max-content (waarde) 263, 340
- max-height (eigenschap) 264
- max-width (eigenschap) 264
- maxlength (attribuut) 164
- Media feature 291
- Media queries 291
 - aspect-ratio 295
 - resolution 294
- Media query 131
 - beeldverhouding 295
 - opbouw 291
 - orientation 294
 - schermresolutie 294
- meta(element) 48
- Metadata 88
- Metadata content 52
- meter (element) 190
- method (attribuut) 156
- Microdata 42
- min (attribuut) 169
- min-content (waarde) 263, 340
- min-height (eigenschap) 264
- min-width (eigenschap) 264
- minlength (attribuut) 164
- minmax() (waarde) 341
- mix-blend-mode (eigenschap) 458
- mp4 139
- Multi-column layout 312
 - aantal kolommen 312
 - balans tussen kolommen 318
 - geen samengevoegde marges 314
 - kolombreedte instellen 313
 - kolommen overspannen 315
 - lijn tussen kolommen 317
 - ruimte tussen de kolommen 316
- multiple (attribuut) 168, 178
- name (attribuut) 159, 163
- nav (element) 57
- Navigatie
 - horizontale balk 416
- Nested browsing context 135
- nonce (attribuut) 42
- none (display) 283
- novalidate (attribuut) 158
- object (element) 138
- Ogg Theora 139
- Ogv 139
- ol (element) 79-80

Onderkop 68
 Ondertiteling 142
 WebVTT 144
 Ongeordende lijst 82
 opacity (eigenschap) 430
 opacity() (filter) 458
 Opmaak
 lettergewicht 401
 lettergrootte 396
 letterstijl 401
 lettertype 395
 lijst 414
 tekst 405
 optgroup (element) 182
 option (element) 182
 order (eigenschap) 346
 Outer display type 281
 output (element) 191
 Overerving 226
 overflow (eigenschap) 273
 overflow-wrap (eigenschap) 413
 Overgang 464
 typen 467
 Overlopende inhoud 273
 Overscroll 275
 overscroll-behaviour (eigenschap) 275
 p (element) 70
 Padding 268
 padding (definitie) 259
 padding (eigenschap) 268
 Paginaheader
 kop 68
 Paginastructuur 50
 Palpabel content 53
 pattern (attribuut) 165
 perspective (eigenschap) 479
 perspective-origin (eigenschap) 480
 Phrasing content 52
 picture (element) 133
 placeholder (attribuut) 168
 PNG 124
 helderheidcorrectie 125
 transparantie 124
 Position
 absolute 300
 fixed 302
 relative 299
 static 299
 sticky 303
 position (eigenschap) 298
 Positionering 297
 absoluut 300
 float 305
 omvattende blok 297
 relatief 299
 Positioneringsschema 297-298
 POST 157
 pre (element) 78
 Prefetch 111
 Preload 111
 progress (element) 189
 Pseudo-element 248
 level 4 250
 Pseudoklasse 244
 doel 245
 dynamisch 244
 gebruikershandeling 245
 ontkenning 248
 taal 246
 UI-toestand 246
 Pseudoklassen
 structurele ~ 246
 px (eenheid) 217
 q (element) 74
 Radiaal kleurverloop 443
 Rand 269
 (half)transparant 270
 korte CSS-notatie 272
 ronde hoeken 445
 Randafbeelding 447
 voorbeeld 449
 Randdikte 269
 Randkleur 269
 Randstijl 271
 Rastercel 328
 Rastercontainer 328
 Rasteritem 328
 Rasterlijn 328
 readonly (attribuut) 167
 Regeleinde 71
 Regelstijl 229
 Reguliere expressie 165
 e-mailadres 171
 postcode 165
 protocol 171
 telefoonnummer 170

Index

rel="dns-prefetch" 112
rel="preload" 111
rel=icon 110
rel=next 111
rel=prefetch 111
rel=prev 111
relative (position) 299
rem (eenheid) 218
required (attribuut) 168
Responsive image 130
RGB 427
RGB-model 119
RGBA 427
right (eigenschap) 299
role (attribuut) 42
Ronde hoek 445
rowspan (attribuut) 198
run-in (display) 282
s (element) 73
Safari 16
samp (element) 75
saturate() (filter) 458
Scalable vector graphics 120
Schaduw 450
Screenreader 285
Secties markeren 54
section (element) 56
Sectioning content 52
 article 55
 aside 58
 nav 57
 section 56
 toepassing 54
select (element) 183
Selector
 * 240
 + 252
 aangrenzend 252
 afstamming 251
 attribuut 243
 elementselector 240
 ID 243
 kind 251-252
 klasse 242
 op hetzelfde niveau 252
 pseudo-elementen 248
 pseudoklassen 244
 punthaak 252
 sibling 251
 spatie 251
 ~ 252
Selectors level 4 253
Semantiek 50
sepia() (filter) 458
Server-side 8
Sidebar 58
Siteroot 118
Sitestructuur 117
 lokale site 118
 root 117
size (attribuut) 167
Sizes (attribuut) 132
Skip-link 78
small (element) 73
source
 src 142
 type 142
source (element) 134, 141
span (element) 76
Specificiteit 224
spellcheck (attribuut) 40
src (attribuut) 127
srcset
 device pixel ratio 131
srcset (attribuut) 131, 134
Stacking context 460
step (attribuut) 169
sticky (position) 303
Stijlblad importen 232
strong (element) 73
style (attribuut) 40, 229
style (element) 229
Stylesheets
 alternatief 109
 sub (element) 75
 Subkop 67
 Subscript 75
 sup (element) 75
 Superscript 75
 SVG 7, 120
 Tab-size (eigenschap) 412
Tabel 196
 afwisselend gekleurde rijen 246
 bijschrift 200
 caption 200
 colgroup 202

colspan 198
 kolommen 198
 lege cellen 203
 rijen 198
 rowspan 198
 structuur 196
 tbody 201
 td 198
 tfoot 201
 th 199
 thead 199
 tr 198
 tabindex (attribuut) 41, 162
 table (display) 282
 table (element) 198
 table-* (display) 283
 Tag 9
 Zie HTML-element
 zelfsluitend 31
 Tagline 68
 target (attribuut) 159
 tbody (element) 201
 td (element) 198
 Tekst 69
 afbreken 413
 alinea 70
 hoofdletters/kleine letters 411
 inspringer 407
 kleur 430
 nieuwe regel 71
 onderstreping 405
 opmaken 405
 responsive 397
 ruimte tussen woorden en letters 412
 schaduw 409
 speciale betekenis 72
 speciale tekens 90
 uitlijnen 405
 witruimte 411
 text-align (eigenschap) 405
 text-decoration (eigenschap) 405
 text-indent (eigenschap) 407
 text-orientation (eigenschap) 408
 text-overflow (eigenschap) 409
 text-shadow (eigenschap) 409
 text-transform (eigenschap) 411
 textarea (element) 186
 tfoot (element) 201

th (element) 199
 thead (element) 199
 Themawisseling 89
 title (attribuut) 41
 title (element) 48
 top (eigenschap) 299
 tr (element) 198
 Track 328, 330
 track (element) 142
 Track-list 340
 transform (eigenschap) 473-474
 transform-origin (eigenschap) 473
 transform-style (eigenschap) 480
 Transformatie 473
 draaien 474
 driedimensionaal 478
 schalen 474
 scheeftrekken 474
 tweedimensionaal 474
 verplaatsen 474
 Transformatiematrix 475
 transition (eigenschap) 464
 transition-timing-function (eigenschap) 467
 translate (attribuut) 41
 Transparantie 430
 transparent (kleurwaarde) 270, 430
 Trapsgewijs opmaakmodel
 Zie CSS
 Tussenruimte 328
 type (attribuut) 134
 type="checkbox" 175
 type="color" 173
 type="date" 175
 type="email" 171
 type="hidden" 180
 type="image" 179
 type="password" 171
 type="range" 172
 type="reset" 179
 type="submit" 178
 type="tel" 170
 type="time" 175
 type="url" 171
 Typeselector 240
 ul (element) 79, 82
 Unicode 91
 Uniform resource locator 116
 Universele selector 240

Index

- URL 116
- usermap (attribuut) 129
- User agent 28
- UTF-8 91
- Validator 19
- value (attribuut) 164
- var() (functie) 485
- Vectorafbeelding 120
- Vendor prefix 211
- Verticaal centreren 266
- Video 138
 - autoplay 140
 - bestandstypen 139
 - codec 139
 - coderen 140
 - controls 140
 - FFmpeg 140
 - HandBrake 140
 - height 140
 - loop 140
 - muted 140
 - ondertiteling 142
 - poster 140
 - preload 140
 - src 140
 - width 140
- video (element) 140
- Videobron 141
- viewport (eenheid) 219
- Viewport (metatag) 295
- visibility (eigenschap) 285, 430
- Visual Studio Code 17
- Visual formatting model 280
- visually-hidden (klasse voor screenreaders) 285
- VP8 139
- VP9 139
- vw, vh, vmin, vmax (eenheden) 219
- W3C 4
- WAI-ARIA 42
- wbr (element) 72
- Web Developer Toolbar 20
- Web Hypertext Application Technology Working Group 5
- Web Open Font Format 387
- Web fonts 387
- WebM 139
- WebP 125
- WebVTT 144
- Webpagina
 - structuur 9
 - vormgeving met CSS 10
- Webserver 8
- Weergavemodel 280
- WHATWG 5
- white-space (eigenschap) 411
- width (attribuut) 128
- width (eigenschap) 260, 263
- WOFF 387
- Woordafbreking 72
- word-break (eigenschap) 413
- word-spacing (eigenschap) 412
- World Wide Web Consortium 4
- writing-mode (eigenschap) 276
- XML 7
- z-index (eigenschap) 302