Frontend programmeren

Frontend programmeren

Mobiele webapps

Versie 2020

Gabriel Sánchez Cano

Boom beroepsonderwijs · Meppel · 2020

Boom beroepsonderwijs
info@boomberoepsonderwijs.nl
www.boomberoepsonderwijs.nl

Auteur: Gabriel Sánchez Cano
Redactie en opmaak: Henk Pel
Titel: Frontend programmeren. Mobiele webapps. Versie 2020
ISBN 978 90 372 5777 9
Vierde druk / eerste oplage

Inhoud

1 Basis HTML5 en CSS3—1

1.1 Inleiding 1
Evolutie van HTML 1
Lay-out en design van een webpagina 1
Van concept tot realisatie 2
Code in het boek 2
De index of homepagina 3
Mappenstructuur van het project aanmaken 3
HTML-editors 4
Geneste elementen 6
Browsers 7
Meta-elementen 8
Semantische elementen 10
Hiërarchie van elementen 12
id of class? 12
1.2 CSS3 14
Inleiding 15
CSS-syntaxis 15
Stijlen coderen 18
<head>-stijlen 18
In-line-stijlen 18
Samenvatting (margin en padding) 22
De stylesheet 24
1.3 Webkleuren 30
1.4 Navigatie 34
Codering van de navigatie 34
Ontwerp van de navigatie 37
1.5 Flexbox 43
Container-properties 44
1.6 Logo 49
Codering van het logo 49
Design van het logo 51
1.7 Hoofd-content 53
Design van de hoofd-content 54
Browser-prefix 55

1.8 Positionering 58
Relatieve positie 59
Lagen 63
1.9 Codering en design van de footer 67
Symbolen in HTML 68
1.10 CSS Grid Layout 70
1.11 De playlist-pagina 76
Codering van playlist-pagina 77
Wat zijn events? 79
1.12 De formulierpagina 83
1.13 De contactpagina 92
Codering van de contactpagina 93
Design van de contactpagina 94
1.14 Material design 95
Materiaaleigenschappen 96
Materiaal met objecten 97
Licht en schaduw van objecten 98
Materiaalomvorming 98
Beweging en betekenis 99
Eigenschappen van objecten en schaduwen 100
Hiërarchie van objecten 100
Interactie 101
Beweging 101
1.15 Een material-design-sjabloon 103
Tabs 105
Codering van de cards 107
1.16 Material-design-project 110
De website XtreemTravel 110
De welkomstpagina 111
De windsurfing-pagina 111
De mountainclimbing-pagina 112
De snorkelenpagina 112

2 Basis JavaScript—113

2.1 Inleiding JavaScript 113
JavaScript-console 115
Taalcomponenten 116
JavaScript-variabelen 118
2.2 Datatypes 123
Het datatype string 124
De methode typeof 124
Het datatype: boolean 125

Het datatype: number (integer) 125
Het datatype: number (floating-point) 125
parseInt() 126
parseFloat() 126
parseInt() 127
toString() 127
2.3 Het Array-object 129
Een Array-object declareren 130
Array-elementen declareren 130
Array-methodes 133
Array-pointers 134
2.4 Objecten 143
2.5 Het Date-object 148
Creëer een Date-object 148
Creëer een Date-object met parameters 149
Creëer een Date-object met string 150
Creëer een Date-object met getallen 150
2.6 Date-methodes 152
getFullYear 152
2.7 Beslissingsstructuren 154
De opdracht if 154
De clausule else 156
&& (AND)-vergelijking 156
|| (OR)-vergelijking 157
De clausule else if 157
De ternary-operator (? :) 158
2.8 Stroomdiagrammen 160
2.9 Switch 163
2.10 Eigen functies 165
Functie of methode 166
Externe functies 168
De tag <script source> 169
Functies dynamisch uitvoeren 172
Scope van variabelen 173
De array arguments[] 174
2.11 Lussen 176
De for()-lus 176
De functie charAt() 179
2.12 De for(in)-lus 182
2.13 De while()-lus 185
De do-while()-lus 188
2.14 Het Math-object 188
Math.abs(x) 189
Math.ceil(x) 189

Math.floor(x) 189
Math.max() 190
Math.min() 190
Math.round() 191
Math.random() 191
2.15 String-methodes 192
indexOf() 193
Length 194
charAt() 195
split() 195
substring() 196
substr() 196
trim() 197
toLowerCase() en toUpperCase() 197
2.16 Algoritmes 198
Het algoritme binary-search 199
Een voorbeeld 199
Pseudocode 200
2.17 Het Document Object Model 205
getElementById 206
DOM-events (gebeurtenis) 206
De functie setTimeout() 208
De functie clearTimeout() 208
Mouse-events 209
Form-events 210
Keyboard-events 214
2.18 Cookies 216
Hoe werk je met cookies? 216
JavaScript met cookies 217
Browser en cookies 220
2.19 Een winkelwagentje in JavaScript 221

3 Gevorderd JavaScript—227

3.1 OOP in JavaScript 227
Class met properties 228
Class met methodes 228
Subclass 228
3.2 Het factory pattern 229
Destructor 231
Object-properties wijzigen 231
3.3 Constructor-pattern 232
3.4 Prototype-pattern 234
Objecten kunnen berichten naar elkaar sturen 235

3.5 Inheritance (overerven) 236
Overschrijven van parent-methodes 238
Delete-function 239
3.6 Een OOP videoPlayer 240
Rekenmachine algoritme 244
3.7 JSON-object-literals 245
Objecten als properties van het object window 247
Geneste JSON-object-literals 247
Chaining properties 248
Literal-objecten bundelen 248
3.8 Speciale functions 250
Externe methodes aan objecten koppelen 250
Call 251
Apply 252
Bind 253
3.9 Anonieme functions 255
Callback-functions 256
Zelfuitvoerende functions 257
Return-functions 257
Function-closures 257
Function-heredity 259
3.10 Lokale opslagcapaciteit (local storage) 261
Local storage lezen 261
3.11 OOP webshop 262
Herbruikbare code 265
3.12 Functions van hogere orde 267

4 TypeScript—273

4.1 Ontwikkelomgeving installeren 273
NPM (Node Package Manager) in Node 274
Installeer TypeScript 274
Installeer Visual Studio Code 274
4.2 Inleiding TypeScript 274
4.3 Objectgeoriënteerd programmeren (OOP) 279
Klassen en objecten 280
UML-klassendiagram 280
Klasse met properties 280
Klasse met methodes 281
Subklasse 281
Constructors 282
Objecten 283
4.4 Access-methodes 285

4.5 Encapsulation (inkapseling) 287
4.6 Eigen methodes 288
4.7 Inheritance 290
Parent-methodes overschrijven 294
Inheritance implementeren 294
4.8 Interfaces 295
Abstracte klasse 297
Object-communication 298
4.9 Foutafhandeling met exceptions 299
De StackTrace 299
4.10 Project OOP in TypeScript 302
4.11 Fullstack TypeScript (frontend & backend) 303
App-opbouw 304
Database verwerken 309
Application Program Interface (API) 311
De Node.js HTTP-module 314
4.12 Graphic User Interfaces (GUI) 316
Views 316
4.13 De webapp FlexTix 318
De homepage 318
De lastminute-pagina 319
De location-pagina 320

Register—325

Oefening baart kunst.
Onbekend

De basis van het spel is om de bal onder controle
te krijgen, zodat je meer tijd hebt om te kijken.
Johan Cruijff

1 Basis HTML5 en CSS3

1.1 Inleiding

HTML is de *lingua franca* (een taal die als gemeenschappelijk communicatiemiddel wordt gebruikt) van het internet. Het is de opmaaktaal die de technologieën achter het internet heeft verenigd en bevordert alle communicatie binnen het internet. Alle communicatie loopt via webpagina's en alle webpagina's zijn in HTML geschreven.

Evolutie van HTML

HTML of HyperText Markup Language is in 1991 ontwikkeld door sir Tim Berners-Lee om wetenschappelijke documenten van het CERN, een Europese organisatie die fundamenteel onderzoek doet naar elementaire deeltjes, toegankelijk te maken. Al snel werden de mogelijkheden van HTML onderkend en in de loop der jaren verschenen steeds betere browsers. Het World Wide Web Consortium (W3C) nam in 1996 de ontwikkeling van HTML over. Versie HTML 3.2 kwam in januari 1997 tot stand. In december 1997 volgde een eerste versie van HTML 4, gevolgd door de eerste versie van HTML5 in januari 2008. Al in 2007 zei Steve Jobs dat HTML5 de software Flash overbodig zou maken. Hij herhaalde zijn woorden bij de introductie van de iPad in 2010.

De belangrijkste aanpassingen in HTML5 zijn het uniform afhandelen van fouten, de mogelijkheid om een document logischer op te bouwen en nieuwe invoertypes. HTML5 wordt ondersteund door alle nieuwe browsers, waaronder FireFox, Google Chrome en Safari.

Lay-out en design van een webpagina

Uit onderzoek blijkt dat de meeste webdesigners in de loop der jaren zijn gaan ontwerpen volgens dezelfde patronen. Bij de meeste moderne websites komen we het volgende ontwerp tegen:

- een webpagina met een banner en navigatie bovenaan;
- een of meer artikelen die de content vormen;
- een kolom aan de rechterkant voor reclame;
- een voettekst met bedrijfsinformatie onderaan.

Een voorspelbare website is makkelijker om in te navigeren en de informatie is eenvoudiger te vinden.

Figuur 1.1 Lay-out van een webpagina

Van concept tot realisatie

In de volgende paragrafen maken we een website voor Radio GaGa. Radio GaGa wil een website met een mooi vormgegeven homepage en met links naar playlists waar de users naar muziek kunnen luisteren of video's van concerten kunnen bekijken. Daarna komt een pagina met een formulier voor het bestellen van tickets voor concerten.

Dit project pakken we op de volgende manier aan.

Ten eerste splitsen we het project op in kleinere delen en ten tweede wisselen we bij ieder deel tussen markup (tags in de vorm van code) en design (ontwerp). In dit hoofdstuk maken we de volgende webpagina's:

- **index.html**
- **playlist.html**
- **formulier.html**
- **contact.html**

Code in het boek

In dit boek staat veel code die ingevoerd moet worden. Het lijkt misschien niet zinvol, maar de enige manier om goed te leren coderen is steeds weer code invoeren, waarna je controleert of je geen fouten hebt gemaakt en daarna kijkt of alles werkt zoals het zou moeten werken – en als dat niet werkt zoek je op waar je een fout hebt gemaakt.

Belangrijk is hierbij de herhaling, waardoor je al doende leert wat alle elementen betekenen.

De index of homepagina

We beginnen met het maken van de indexpagina. De eerste pagina die je maakt krijgt altijd de naam index.html (of index.php, index.asp enzovoort). De webserver zoekt bij een pagina-request (het opvragen van een pagina via de webbrowser) van het volledige domein (bijvoorbeeld http://www.domeinnaam.nl) altijd naar een indexpagina, tenzij er een andere url (Universal Resource Locator) wordt ingetypt zoals: http://www.domeinnaam.nl/contact.html. Dan gaat de webserver op zoek naar contact.html.

De homepagina is de index.html-pagina en moet er als volgt uitzien:

Figuur 1.2 De indexpagina

Deze pagina krijgt bovenaan een navigatiebar met daarin de navigatie naar de andere pagina's. Daaronder komt het logo van Radio GaGa. Onder het logo komt de content. De content bestaat uit een draaiende disk. Links en rechts, in het midden van de pagina, komen bladerknoppen met een simpele navigatie naar de volgende of de vorige pagina. Als laatste maken we de footer van de pagina met links naar Twitter, YouTube en Facebook van Radio GaGa.

Mappenstructuur van het project aanmaken

Een gestructureerde aanpak betekent een duidelijke mappenstructuur voor ons project zodat we precies weten waar onze HTML en content te vinden is. De onderstaande figuur toont de gewenste structuur. Maak op je computer de mappenstructuur aan zoals hieronder. De map **RadioGaGa** bevat dus de mappen **images** en **videos**.

Figuur 1.3

HTML-editors

Om de opgaven en Vaardigheid-lab-opdrachten uit dit boek te kunnen maken heb je een HTML-editor nodig. Een HTML-editor is een applicatie voor het schrijven van HTML-teksten. Deze teksten noemen we webpagina's. Hier zijn twee websites waar je gratis HTML-editors kunt downloaden:

- **Brackets** download je vanaf de volgende website: http://brackets.io/?lang=en
- **Sublime text** download je vanaf de volgende website: http://www.sublimetext.com
- **Visual Studio Code** download je vanaf https://code.visualstudio.com/download

Na het downloaden en installeren van je favoriete editor begin je met het coderen van de indexpagina.

In deze paragraaf maken we de code van de indexpagina. We maken kennis met de volgende HTML-elementen:

- comments
- DOCTYPE
- geneste elementen
- meta-elementen
- semantische elementen
- id en classes

Aan het einde van deze paragraaf laat je de volgende opgaven door je docent controleren:

Planning	Inleveren
	Webbouw 1 t/m 4
	Vaardigheid-lab 01

HTML-elementen

Webpagina's bestaan uit platte tekst met *markeringstekens*, ook wel aangeduid als *tags*. Hierin wordt de lay-out aangegeven, oftewel hoe de tekst gestructureerd wordt. De *tags* worden door de webbrowser geïnterpreteerd en de gebruiker krijgt de opgemaakte tekst in de vorm van een webpagina te zien. Een markering tussen haakjes, bijvoorbeeld `<html>` of `<form>`, wordt een *tag* genoemd. De syntaxis van tags is als volgt:

```
<open tag> content van de tag </sluiten tag>
```

Syntaxis betekent de opbouw en structuur van je code volgens de aangegeven regels.
Als je een *begintag* hebt geopend, bijvoorbeeld `<html>`, eindig je *altijd* met een *eindtag*, bijvoorbeeld `</html>`. Je ziet dat de *eindtag* altijd een / bevat.

We beginnen nu met het coderen van onze eerste webpagina: de indexpagina.

- ***Webbouw 1***
  Open je favoriete HTML-editor en typ de volgende code. Sla het bestand op als **index.html** in de map **RadioGaGa**.

```
<!-- website RadioGaGa -->
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta http-equiv="Content-Type"
    content="text/html;
    charset=UTF-8">
  <title>RadioGaGa</title>
</head>
<body>
  ...
</body>
</html>
```

Op de puntjes coderen we verder de content (dat wat op de webpagina zichtbaar moet zijn) van de pagina tussen de tags `<body>` en `</body>` in.

Hier volgt een uitleg over de elementen in Webbouw 1.

Element <!-- commentaren -->

We begonnen de code met het commentaar:

```
<!-- website RadioGaGa -->
```

Zulk soort commentaar tussen `<!--` en `-->` kunnen we overal op de pagina plaatsen en is bedoeld om de structuur en code van de pagina toe te lichten. Het geeft iemand die jouw code leest een toelichting op de keuzes die je hebt gemaakt. Het is ook voor jou een hulpmiddel als je op een later moment je code nog een keer doorleest. Commentaar heeft geen effect op de weergave van de pagina.

Element <!DOCTYPE html>

Browsers kunnen meerdere soorten documenten weergeven, zoals webpagina's en PDF's. Het is daarom belangrijk om ons HTML-document te identificeren met de tag `<!DOCTYPE html>`. Zo kan de browser onze code vertalen in een webpagina.

Element <html lang="nl">

De tweede regel in Webbouw 1 is het element `<html>`. We coderen de namen van alle tags in kleine letters. De enige tag met hoofdletters is het element `<!DOCTYPE>`. Sommige elementen kunnen attributen bevatten. Alle attributen krijgen een naam en een waarde tussen dubbele aanhalingstekens. In dit geval is het attribuut `lang` (language) `lang="nl"`. Zo geef je aan dat de content van deze pagina in het Nederlands wordt geschreven.

Element </html> sluiten

Zoals te zien is sluiten we alle elementen met het slash-teken (/). Alle elementen moeten gesloten worden. Als je een element niet goed sluit, leidt dit tot fouten in het weergeven van de pagina.

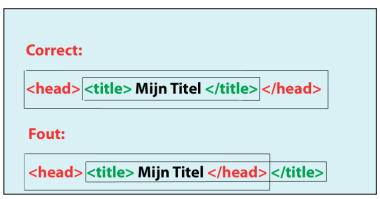
Geneste elementen

Het eerste wat opvalt in Webbouw 1 zijn de geneste elementen. Dat zijn elementen binnen andere elementen. In dit geval bevat `<html>` de elementen `<head>` en `<body>`. Het `<head>`-element bevat op zijn beurt de `<meta>`- en `<title>`-elementen. Een genest element begin je met de Tab-toets. Bijvoorbeeld:

```
<html lang="nl">
<head>
  <meta http-equiv="Content-Type"
    content="text/html;
    charset=UTF-8">
  <title>RadioGaGa</title>
</head>
<body>
  ...
</body>
</html>
```

We gebruiken tabs om de code leesbaarder te maken. Zo zien we in één oogopslag welke elementen andere geneste elementen bevatten. Je kunt zo zien dat de begintag `<html>` helemaal onderaan wordt gesloten met de eindtag `</html>`. Binnen de begintag en eindtag bevinden zich de child-elementen `<head>` en `<body>`. Van `<head>` en `<body>` is de `<html>`-tag het parent-element. Het gebruik van tabs heeft geen effect op het weergeven van de webpagina.

Geneste elementen moeten in de juiste volgorde worden gesloten. Een genest element moet als eerste gesloten worden. Daarna sluit je het element van het niveau hoger. De volgende figuur laat zien hoe je geneste elementen goed sluit.



Figuur 1.4 Geneste elementen

Element <head>

We hebben de webpagina ingedeeld in `<head>` (hoofd) en `<body>` (lichaam). We gebruiken het `<head>`-element voor meta-informatie en voor andere elementen zoals `<title>`, en het `<body>`-element voor de content.

Element <title>

Iedere pagina krijgt een titel. De volgende regel zorgt er bijvoorbeeld voor dat de titel `RadioGaGa` bovenaan de browser verschijnt:

```
<title>RadioGaGa</title>
```

De titel `RadioGaGa` verschijnt, maar niet de `<title>`-tags.

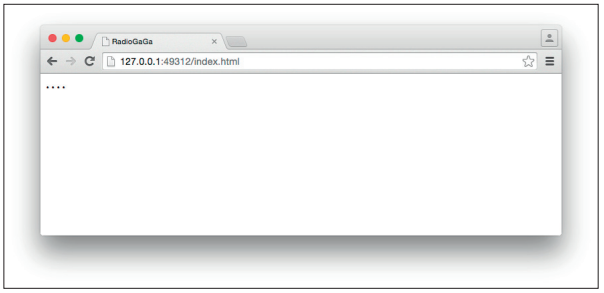
Na het opslaan van Webbouw 1 als **index.html** in de map **RadioGaGa** controleren we het resultaat in meerdere browsers.

Browsers

Om de resultaten van de opgaven en de vaardigheid-lab-opdrachten te kunnen zien moeten we eerst de volgende browsers downloaden en installeren.

Browser	Website
	https://www.google.nl/chrome/browser/desktop/
	https://support.mozilla.org/nl
	http://www.opera.com/
	http://safari.nl.softonic.com/
	http://windows.microsoft.com/nl-nl/internet-explorer/download-ie

Na het downloaden en installeren van deze browsers openen we het bestand **index.html** door er met de muis op te klikken. De browser presenteert de code van Webbouw 1 zoals te zien is in de volgende figuur.



Figuur 1.5 Het resultaat van Webbouw 1

LET OP
Gebruik dubbele aanhalingstekens (") in plaats van twee enkele aanhalingstekens (' '). Bijvoorbeeld:

`<html lang=''nl''>` is niet correct.
`<html lang="nl">` is correct.

We krijgen alleen maar de puntjes binnen het `<body>`-element te zien. In ons geval hebben we Chrome gebruikt om de pagina te bekijken. Het is dus niet nodig om een eigen domeinnaam bij een provider aan te schaffen om de resultaten van de opgaven te kunnen zien. Met de browsers kunnen we zowel internetten als onze webpagina's bekijken. Als je later je eigen website wilt uploaden moet je wel een eigen domeinnaam aanschaffen.

Meta-elementen

We gebruiken `<meta>`-elementen voor het toevoegen van meta-informatie. Dit is informatie over de informatie van de pagina. Deze informatie is relevant voor browsers en zoekmachines. Google en Yahoo maken gebruik van deze informatie in hun zoekalgoritmes. We kunnen meta-elementen ook gebruiken voor andere doelen, zoals het automatisch laden van een andere pagina. Ieder meta-element bevat de attributen:

```
name
content
```

Dat zijn de naam en de content van het meta-element. Bijvoorbeeld, om de naam van de auteur van de pagina aan te duiden codeer je het volgende:

```
<meta name="author" content="naam van auteur">
```

In de volgende opgave coderen we belangrijke meta-informatie over de te bouwen website.

Een meta-tag hoef je niet te sluiten met een / (het slash-teken).

LET OP
We mogen een lange HTML-tag alleen afbreken na een spatie. Dit doen we bijvoorbeeld om de code te verduidelijken.

- ***Webbouw 2***

Open **index.html** en voeg de volgende meta-informatie eraan toe.
Let op! De code voeg je dus toe tussen `<head>` en `</head>`. Deze andere codes heb je in Webbouw 1 al getypt.

```
<head>
  <meta http-equiv="Content-Type"
    content="text/html;
    charset=UTF-8">
  <meta name="robots" content="all">
  <meta name="language" content="Dutch">
  <meta name="author" content="je eigen naam">
  <meta name="description" content="RadioGaGa marketing">
  <meta name="keywords" content="RadioGaGa, muziek, playlist">
  <meta name="copyright" content="copyright">
  <title>RadioGaGa</title>
</head>
<body>
  ...
</body>
</html>
```

Hieronder volgt een uitleg over Webbouw 2.

<meta charset>

`Charset` is de set tekens ontworpen voor computers. In Webbouw 2 hebben we het volgende attribuut gespecificeerd:

```
charset="UTF-8"
```

Dit is de karakterset voor alle Europese talen. Maar er zijn andere karaktersets, bijvoorbeeld:

charset	taal
`charset="ISO-8859-6"`	Grieks
`charset="ISO-8859-7"`	Arabisch
`charset="UTF-8"`	Europese talen

<meta name="robots">

Hoe vertel je de spider van een zoekmachine dat je wilt dat hij alleen de eerste pagina, of de hele website indexeert? Dat doe je met de zogenaamde `robots` metatag. Om de pagina in zijn geheel te indexeren kies je voor `all`.

```
<meta name="robots" content="all">
```

Om alle robots te verhinderen een pagina van je site te indexeren, moet je de volgende metatag in het <head>-gedeelte van de pagina plaatsen:

```
<meta name="robots" content="noindex">
```

<meta name="language">

Om de taal van de content specifiek aan te geven gebruiken we dit attribuut:

```
<meta name="language" content="Dutch">
```

<meta name="author">

Om de naam van de auteur van de pagina aan te geven gebruiken we:

```
<meta name="author" content="je eigen naam">
```

<meta name="description">

Voor een korte beschrijving van de website gebruiken we het attribuut `description`:

```
<meta name="description" content="RadioGaGa marketing">
```

<meta name="keywords">

Voor een lijst met sleutelwoorden over de website gebruiken we het attribuut `keywords`:

```
<meta name="keywords" content="RadioGaGa, muziek, playlist">
```

<meta http-equiv="refresh">

Het attribuut `http-equiv="refresh"` kunnen we gebruiken om het aantal seconden voor het *refreshen* van deze pagina te specificeren. Het kan ook de naam van een pagina aangeven waar we na een aantal seconden automatisch naar willen doorlinken, bijvoorbeeld een inleidingpagina:

```
<meta http-equiv= "refresh"content="10;url=http://website.com">
```

Semantische elementen

De volgende HTML5-elementen hebben een semantische betekenis die de positie en het doel van het element in een webpagina beschrijven.

`<head>`	hoofd
`<body>`	lichaam
`<nav>`	navigatie
`<main>`	hoofd-content
`<section>`	sectie
`<div>`	divisie
`<p>`	paragraaf
`<footer>`	voet

Element <body>

Na het definiëren van het <head>-element met alle <meta>- en <title>-elementen kunnen we verder met het coderen van de structuur van de pagina. Dit doen we binnen het <body>-element.

Element <nav>

Binnen het <nav>-element coderen we de elementen die nodig zijn voor de navigatie van de website.

Element <main>

In het <main>-element coderen we de hoofd-content van de pagina, bijvoorbeeld een groep afbeeldingen en teksten die het thema van de pagina definiëren.

Element <section>

Het <main>-element mag één of meer secties of kolommen hebben. Dit doen we met het <section>-element. Hier kunnen we verschillende gerelateerde content met een thema combineren. Bijvoorbeeld, een <section>-element met het thema sport mag een sectie sport en een sectie sportnieuws hebben.

Element

Binnen het element plaatsen we generieke (diverse) content, bijvoorbeeld algemene plaatjes en teksten. Een <section> mag meerdere 's hebben.

Element <footer>

Het <footer>-element is bedoeld voor informatie over het bedrijf, de webdesigner en de copyrights. Ook kunnen we hier belangrijke links vinden.

- *Webbouw 3*

  Open **index.html** en vervang de puntjes door de code binnen de <body>-tags:

```
<body>
  <nav>
    <h3>Navigatie</h3>
  </nav>
  <div id="logo">
    <h3>Logo</h3>
  </div>
  <main>
    <h4>Hoofd-content</h4>
  </main>
  <footer>
    <h3>Footer</h3>
  </footer>
</body>
</html>
```

Hiërarchie van elementen

Parent-elementen bevatten child-elementen en dit zijn geneste elementen. Het `<html>`-element is het parent-element van alle andere elementen. Het `<body>`-element is een child-element van het `<html>`-element maar ook een parent-element van de elementen `<nav>`, `<main>` en `<footer>`.

```
<!-- website RadioGaGa -->
<!DOCTYPE html>
<html lang="nl">
    <head>
        <meta http-equiv="Content-Type"
          content="text/html;
          charset=UTF-8">
        <meta name="robots" content="all">
        <meta name="language" content="Dutch">
        <meta name="author" content="je eigen naam">
        <meta name="description"
        content="RadioGaGa marketing">
        <meta name="keywords"
        content="RadioGaGa, muziek, playlist">
        <meta name="copyright" content="copyright">
        <title>RadioGaGa</title>
    </head>
    <body>
        <nav>
            <h3>Navigatie</h3>
        </nav>
        <div id="logo">
            <h3>Logo</h3>
        </div>
        <main>
            <h3>Hoofd content</h3>
        </main>
        <footer>
            <h3>Footer</h3>
        </footer>
    </body>
</html>
```

Figuur 1.6 Parent- en child-elementen

Elementen <h1> tot <h6>

Met de elementen `<h1>` tot `<h6>` definiëren we de lettergrootte van teksten binnen de elementen. Bijvoorbeeld:

```
<h1>RadioGaGa</h1>
<h2>RadioGaGA</h2>
<h3>RadioGaGa</h3>
<h4>RadioGaGa</h4>
<h5>RadioGaGA</h5>
<h6>RadioGaGA</h6>
```

id of class?

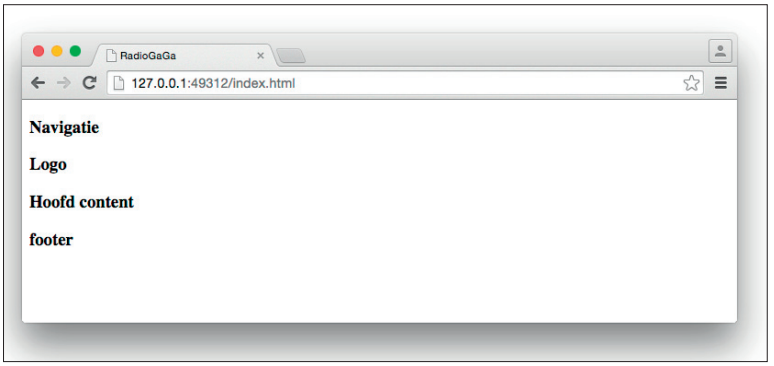
Als je een element identificeert met het attribuut `id=`, dan maak je hiervan een uniek element. We mogen geen twee elementen met dezelfde id benoemen. We maken een element uniek zodat hieraan een bepaalde opmaak kunnen toekennen. In Webbouw 3 hebben we het element:

```
<div id="logo">
```

Wanneer je meerdere elementen dezelfde visuele attributen wilt geven, krijgen ze allemaal dezelfde class. Bijvoorbeeld:

```
<div id="logo" class="blauw">
<div id="footer" class="blauw">
```

Zo identificeren we elementen om ze later visuele attributen te kunnen geven. Dat doen we in de volgende paragraaf wanneer we beginnen met het coderen van CSS-stylesheets.



Figuur 1.7 Resultaat van Webbouw 3

LET OP
Als je geen resultaat krijgt dan moet je je code verbeteren. De meest voorkomende beginnersfouten zijn het niet goed afsluiten van de tags.
Ook spellingsfouten bij namen van de elementen zijn een probleem. Bijvoorbeeld `<budy>` in plaats van `<body>`.
Of `<html>` met het getal `1` in plaats van `<html>` met de letter `l`.
Het resultaat van deze fouten kan een lege pagina in de browser zijn.

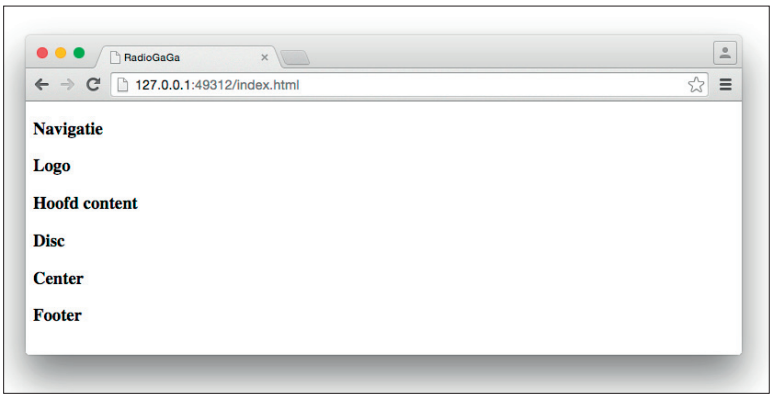
- ***Webbouw 4***
  Open **index.html** en voeg op een correcte manier het nieuwe `<div>`-element binnen het `<main>`-element toe:

```
<div id="disc"> <h4>Disc</h4>
```

Binnen `disc` voeg je op de juiste manier het volgende element toe:

```
<div id="center"> <h4>Center</h4>
```

Het resultaat moet er als volgt uitzien:



Figuur 1.8 Resultaat van Webbouw 4

- *Vaardigheid-lab 01*
  In deze vaardigheid-lab-opdracht maak je een nieuw script en sla je het op als **labs.html**. In dit script maak je vier div's met de volgende hiërarchie:

```
div 1
div 2
   div 3
      div 4
```

1.2 CSS3

In deze paragraaf beginnen we met de vormgeving van de indexpagina. We maken kennis met de volgende CSS-attributen:

- width
- height
- commentaren
- background
- overflow
- text-align
- margin
- border
- font
- box-shadow

Aan het einde van deze paragraaf laat je de volgende opgaven door je docent controleren:

Planning	Inleveren
	Oefening 1 t/m 6 Webbouw 5 t/m 6
	Vaardigheid-lab 02

CSS (Cascading Style Sheets) is ontstaan uit onvrede onder webdesigners over het gebruik van HTML-tags voor het opmaken van webpagina's. De CSS Working Group van de W3C heeft versie 1 van CSS al eind 1996 ontwikkeld. Het gaf designers voor het eerst de mogelijkheid om de opmaak van lettertypen en de positie van de elementen van een webpagina te bepalen.

Inleiding

CSS3 biedt meer geavanceerde mogelijkheden en dringt het aantal Javascript-elementen terug, waardoor de totale omvang van een pagina kleiner wordt, wat ten goede komt aan de ranking bij zoekmachines. Belangrijk zijn daarnaast de grafische mogelijkheden, zoals drop shadows, ronde hoeken, meerdere achtergronden, animaties en transparanties.

CSS-syntaxis

Met CSS-commando's kunnen we één of meer HTML-elementen selecteren om vervolgens stijlattributen toe te wijzen. Zo kunnen we de geselecteerde elementen visuele eigenschappen toeschrijven. Elk commando heeft de volgende syntaxis:

```
selector {attribuut: waarde;}
```

Het commando bestaat uit twee delen:

`selector` {`attribuut:waarde` ;}

1. `Selector` is de naam van het HTML-element dat we willen selecteren.
2. Het paar `attribuut:waarde` beschrijft het attribuut en de attribuutwaarde die we willen toekennen aan het geselecteerde HTML-element.

We schrijven de naam van het HTML-element zonder de haakjes < >. Tussen de accolades { } schrijven we het paar `attribuut:waarde`. We eindigen met een puntkomma. We kunnen de geselecteerde elementen een of meer attributen toekennen, bijvoorbeeld:

```
h1 {
  attribuut1: waarde1;
  attribuut2: waarde2;
}
```

Elementen selecteren

Hier zien we een voorbeeld van het selecteren van meerdere HTML-elementen om vervolgens visuele attributen toe te wijzen.
We passen de syntaxis toe:

```
h1 {
  color: black;
  text-align: center;
}
```

Color en text-align

In het vorige voorbeeld hebben we alle <h1>-elementen geselecteerd om vervolgens het attribuut color (tekstkleur) toe te wijzen. Het attribuut color heeft de waarde black (zwart) gekregen. Het resultaat is dat de content van alle <h1>-elementen zwart wordt weergegeven. Het attribuut text-align (tekst uitlijnen) krijgt de waarde center (centreren). De attributen color en text-color zijn verwisselbaar en worden herkend door alle browsers.

Een #id selecteren

Om een uniek element te selecteren gebruiken we de id van het element. Bijvoorbeeld, in **index.html** is er het element . We kunnen door middel van de id dit specifieke element selecteren. In css selecteer je een id door een # te plaatsen met daarachter de naam van de id.
Hier passen we de syntaxis toe:

```
#logo {
  height: 200px;
}
```

Omdat geen twee elementen dezelfde id mogen hebben, hebben we hier het specifieke element met id="logo" geselecteerd en het attribuut height (hoogte) toegekend met de waarde 200px, met andere woorden: 200 pixels hoog.

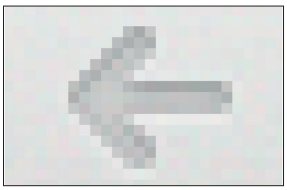
Een .class selecteren

Het kan ook zijn dat je meerdere elementen van dezelfde opmaak wilt voorzien. We hebben gezien dat een id wordt toegekend aan een uniek element. Een class gebruiken we om van meerdere elementen een groep te maken die wij van dezelfde opmaak kunnen voorzien.
Om alle elementen met dezelfde class-naam te selecteren gebruiken we een punt gevolgd door de naam van de class. We kunnen door middel van de class-naam visuele attributen voor meerdere elementen definiëren.
Hier passen we de syntaxis toe:

```
.blauw {
  height: 200px;
}
```

Wat zijn pixels?

Een pixel is een vierkante punt op het beeldscherm van je PC. De resolutie van beeldschermen wordt weergegeven in aantallen pixels. Schermen van smartphones hebben bijvoorbeeld 600 pixels in de breedte en schermen van pc's hebben tot 1200 pixels in de breedte. Pixels zijn erg klein. Om de pixels van een beeld te zien moeten we de afbeelding vergroten. In de volgende figuur zien we de pixels van de afbeelding.



Figuur 1.9

width en height

Met de attributen `width` (breedte) en `height` (hoogte) definiëren we de dimensies van de elementen in pixels en percentages van het beeldscherm, bijvoorbeeld:

```
#logo {
  width: 100%;
  height: 200px;
}
```

color

Om te werken met tekst in kleur gebruiken we het attribuut `color` als volgt:

```
color: black;
```

background-color

Om een achtergrondkleur toe te passen gebruiken we het attribuut `background-color` als volgt:

```
background-color: red;
```

Hier volgt een lijst met de meest voorkomende webkleuren – waarbij je ook het gebruik van camelCase ziet:

```
aqua
aquaMarine
black
blue
blueViolet
brown
cadetBlue
chocolate
coral
crimson
darkBlue
gray
green
greenYellow
indigo
orangeRed
red
```

border

Borders zijn de randen van de elementen. Met het attribuut `border` bepalen we de stijl, kleur en dikte van de borders. Er zijn de volgende stijlen:

`border: solid`	ononderbroken
`border: dotted`	puntjes
`border: dashed`	streepjes

Bijvoorbeeld de volgende borderstijlen:

```
border: 2px dotted black;
background-color: orange;
```

geven een zwarte rand van puntjes van 2 pixels dik met een oranje achtergrond, zoals hierna:

Stijlen coderen

Er zijn drie manieren om stijlen te declareren:

1. Binnen het `<head>`-element (voor korte opdrachten)
2. In line (voor het direct toepassen van stijlen)
3. In een externe style sheet (voor grotere opdrachten)

<head>-stijlen

Stijlen binnen de `<head>` coderen we met de tags `<style>` en `</style>`, bijvoorbeeld:

```
<head>
  <style>
    body{
      width: 40%;
      font-size: 22px;
    }
  </style>
</head>
```

Hier hebben we stijlen voor de `<body>` gecodeerd.

In-line-stijlen

In-line-stijlen coderen we direct binnen een element met de eigenschap `style=`, bijvoorbeeld:

```
<h1 style="color: green;">
```

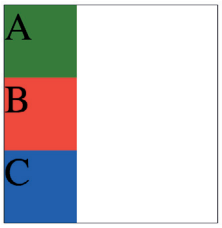
Hier zeggen we dat de tekstkleur in element `<h1>` groen is.

In de volgende opgave passen we twee methodes toe bij het coderen van stijlen.

Hier volgt een aantal extra oefeningen om de CSS-attributen goed te doorgronden

- *Oefening 1: background-color*
  Open een nieuw bestand en sla het op als **oefening.html**. Voeg de volgende code toe:

```
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta http-equiv="Content-Type" content="text/html;
  charset=UTF-8">
  <title>Oefening</title>
  <style>
    body {
      border: 1px solid;
      width: 40%;
      font-size: 80px;
    }
    #a {
      width: 150px;
      height: 150px;
      background-color: green;
    }
    #b {
      width: 150px;
      height: 150px;
      background-color: red;
    }
    #c {
      width: 150px;
      height: 150px;
      background-color: blue;
    }
  </style>
</head>
<body>
  <div id="a">
    A
  </div>
  <div id="b">
    B
  </div>
  <div id="c">
    C
  </div>
</body>
</html>
```



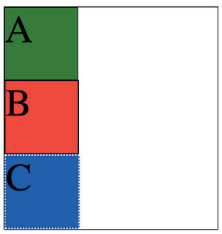
Figuur 1.10 Resultaat van oefening 1

- *Oefening 2: Borders*
  In deze oefening coderen we in-line-stijlen, bijvoorbeeld:

```
<div style="border: 1px solid black;">
```

  voor de borders van elementen A, B en C:

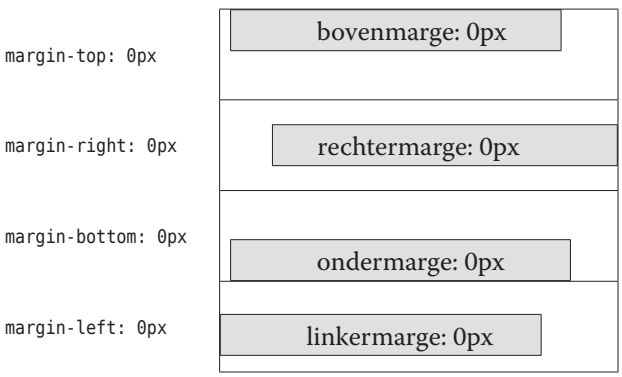
- A met een ononderbroken zwarte rand met een dikte van 1 pixel.
- B met een ononderbroken zwarte rand met een dikte van 2 pixels.
- C met een ononderboken en gestippelde rand met een dikte van 3 pixels.



Figuur 1.11 Resultaat met borders

margin

Marges zijn de ruimtes tussen alle elementen. We kunnen individuele marges specificeren:



Hierboven zien we de marges van de grijze elementen ten opzichte van de elementen waar ze zich in bevinden.

We kunnen ook alle marges in één keer vaststellen. Dit doen we met de klok mee: boven, rechts, onder, links:

```
margin: 0 0 0 0;
```

Als alle vier de marges hetzelfde zijn doe je dit zo:

```
margin: 0; of margin: 5%;
```

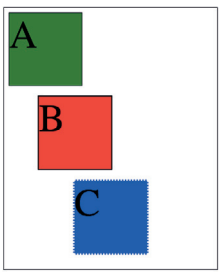
- *Oefening 3: Marges*
  In deze oefening gaan we meer stijlen toevoegen binnen de `<head>`.

```
<head>
  <style>
     ...
  </style>
</head>
```

A krijgt de volgende marges:
- Een bovenmarge van 10 pixels.
- Een linkermarge van 10 pixels.

B krijgt de volgende marges:
- Een bovenmarge van 20 pixels.
- Een linkermarge van 70 pixels.
- Een ondermarge van 20 pixels.

C krijgt de volgende marges:
- Een auto linkermarge.
- Een auto rechtermarge.
- Een ondermarge van 30 pixels.



Figuur 1.12 Resultaat met marges

- *Oefening 4: Padding*
  De `padding` (invulling) verwijst naar de ruimte tussen de rand en de content van een element. Een `padding` geven we aan in percentages of pixels. Bijvoorbeeld:

```
padding: 10px 5px 0% 30px;
```

is hetzelfde als:

```
padding-top: 10px;
padding-right: 5px;
padding-bottom: 0%;
padding-left: 30px;
```

Wanneer alle `paddings` gelijk zijn zeggen we gewoon:

```
padding: 5%;
```

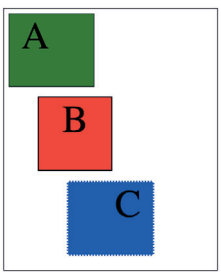
Wanneer je bij padding twee waarden meegeeft, bijvoorbeeld:

```
padding: 10px 5px;
```

dan is de eerste waarde voor `top` en `bottom` (`padding 10px`), en de tweede waarde voor `right` en `left` (`padding 5px`).

Codeer in de `<head>` de volgende stijlen voor A en C.

- A met een linker-padding van 25 pixels.
- C met een rechter-padding van 25 pixels.

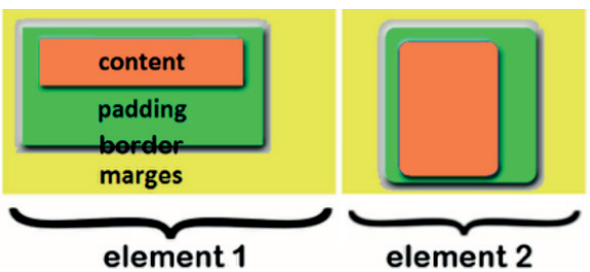


Figuur 1.13 Resultaat met paddings

In deze figuur zien we dat elementen A en C breder dan B zijn geworden. Dat komt door de toegevoegde paddings. Het resultaat is dat A en C nu 175 pixels breed zijn.

Samenvatting (margin en padding)

Het attribuut `padding` is de ruimte tussen de content en de border van het element. Het attribuut `margin` is de ruimte tussen elementen.



Figuur 1.14

- *Oefening 5: Box-shadow*
  Voor schaduwen van elementen gebruiken we box-shadow. Bijvoorbeeld:

```
box-shadow: 5px 5px 10px black;
```

Hier coderen we een zwarte schaduw van:

- 5 pixels rechts
- 5 pixels onder
- 10 pixels straal

Met negatieve getallen kunnen we experimenteren met negatieve schaduwen, bijvoorbeeld:

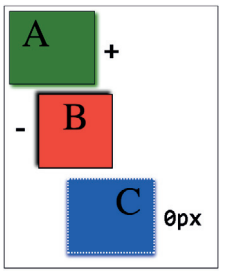
```
box-shadow:-5px -5px 10px black;
```

Dit is een schaduw met:

- 5 pixels links
- 5 pixels boven

Codeer in de <head> schaduwstijlen voor A, B en C.

- A met een groene schaduw van 5px bij 5px en met een straal van 10px.
- B met een zwarte schaduw van -5px bij -5px (negatieve) en met een straal van 10px.
- C met een blauwe schaduw van 0px bij 5px y (onder) en met een straal van 10px.



Figuur 1.15 Resultaat met schaduwen

- *Oefening 6: Tekstschaduw*
  We kunnen ook schaduwen coderen voor teksten. Bijvoorbeeld:

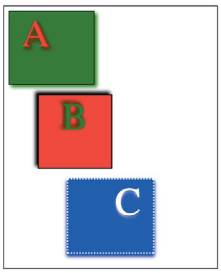
```
box-shadow: 5px 5px 10px black;
```

Dit is een positieve schaduw voor teksten. Het lijkt alsof het licht van boven links komt. Met negatieve getallen kunnen we met negatieve schaduwen experimenteren. Bijvoorbeeld:

```
box-shadow:-5px -5px 10px black;
```

Codeer in de <head> schaduwen voor de teksten A, B en C.

- A met rode tekst en een rode schaduw
- B met groene tekst en een negatieve groene schaduw
- C met witte tekst en een witte schaduw onder



Figuur 1.16 Resultaat met tekstschaduw

Hier merken we op dat de schaduw van de letters dezelfde richting uitgaan, net als de schaduwen van de borders.

De stylesheet

In de volgende opgave maken we onze eerste stylesheet met de naam **styles** en met de extensie **.css**. Alle stylesheets krijgen de extensie **.css** dat staat voor Cascading Style Sheet. We gebruiken stylesheets om de structuur (HTML) en de opmaak (CSS) te scheiden. Eenmaal gecodeerd kunnen we de stijlen uit een stylesheet hergebruiken.

- ***Webbouw 5***

  Codeer de volgende stijlen en sla ze op als **styles.css** in dezelfde map als **index.html**. De uitleg volgt later.

```
/* Stijlen voor RadioGaGA */
html {
  height: 100%;
  width: 100%;
  background-color: lightgray;
  overflow: auto;
}
body {
  width: 96%;
  margin-top: 0;
  margin-left: auto;
  margin-right: auto;
  margin-bottom: 5%;
  border: 1px solid;
  overflow: inherit;
  background-color: #fff;
  text-align: center;
  font-family: Verdana, Arial, sans-serif;
  color: #000;
  font-size: 16px;
  font-style: normal;
}
h3 {
  margin: 0;
  box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.5);
}
```

De <link>-tag

Om onze webpagina te verbinden met onze stylesheet gebruiken we de tag `<link>`. Door CSS op te nemen in een apart bestand breng je een scheiding aan tussen de opmaak en de structuur van een pagina. Tevens kun je op elke gewenste pagina een koppeling maken met het CSS-bestand, zodat die pagina de opmaak krijgt van het CSS-bestand. Heel handig dus omdat je hierdoor maar op één plaats je opmaak hoeft te wijzigen in plaats van op elke webpagina.
In de volgende opgave linken we **index.html** met het bestand **stijlen.css**.

- ***Webbouw 6***

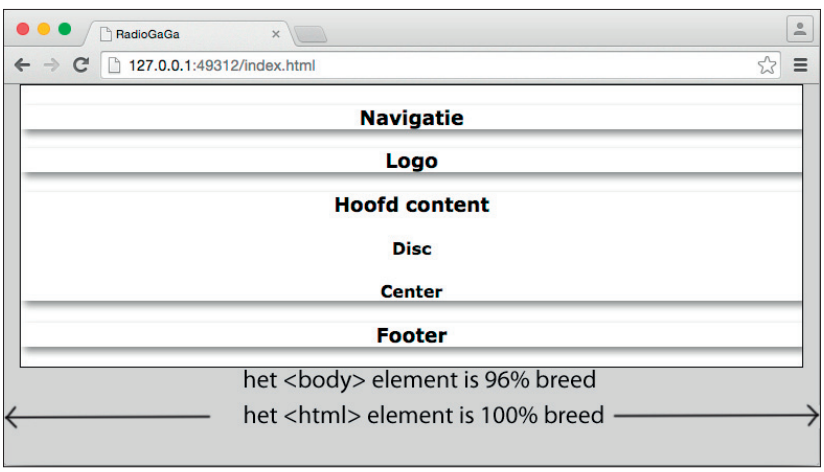
Open **index.html** en voeg de volgende `<link>`-tag eraan toe. Dit moet gebeuren binnen je `<head>`-element. Zorg ervoor dat de elementen een schaduw krijgen zoals te zien is in figuur 1.17.

```
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="Content-Type"
    content="text/html;
    charset=UTF-8">
  <link rel="stylesheet" href="styles.css">
  <title>RadioGaGa</title>
</head>
```

De tag `<link>` bevat de volgende attributen:

`rel="stylesheet"`	Stelt de relatie vast met `stylesheet`
`href="styles.css"`	Verwijst naar het bestand `styles.css`

Open **index.html** met een browser. Als alles goed is verlopen zie je de volgende figuur.



Figuur 1.17

Als je andere resultaten krijgt, kijk dan nogmaals naar je code en je stijlen en zie of je fouten kunt verbeteren. Zorg ervoor dat de `<link>` tussen je twee bestanden goed werkt.

Breedte in percentages

Het is raadzaam om percentages te gebruiken voor de breedte van je elementen. Zo zorg je ervoor dat als het browservenster kleiner wordt de breedte van je `<body>`-element altijd `96%` van het venster blijft. We zeggen dan dat de breedte van je elementen elastisch is.

Hieronder zie je een `<body>`-element met een vaste breedte van `1400px`. Een element met een vaste breedte schikt zich niet naar de breedte van het venster.

Figuur 1.18

Hier volgt een uitleg van de stijlen in Webbouw 5.

/* commentaar */

Met commentaar kunnen we onze stijlen beschrijven. CSS-commentaar plaatsen we tussen `/*` en `*/` in. Bijvoorbeeld:

```
/* Stylen voor RadioGaGa */
```

background-color

Het `<html>`-element heeft een grijze achtergrond gekregen met:

```
background-color: lightgray;
```

overflow

Als een element te klein is voor de tekstinhoud dan zeggen we dat de tekst overloopt. In deze gevallen kunnen we kiezen voor het tonen van een scrollbar (schuifbalk).

```
overflow: auto;
```

text-align

Je ziet in het resultaat van Webbouw 5 en 6 dat alle teksten gecentreerd zijn weergegeven. Het attribuut `text-align` (tekst uitlijnen) kent vier opties:

`text-align: left`	links uitlijnen
`text-align: center`	centreren
`text-align: right`	rechts uitlijnen
`text-align: justify`	de tekst uitvullen over de hele regel

Een tekst centreren is niet hetzelfde als het centreren van een element. In ons geval hebben we het `<body>`-element gecentreerd binnen het `<html>`-element. Dit hebben we gedaan met het attribuut `margin`.

Elementen centreren

Ons `<body>`-element heeft in Webbouw 5 een bovenmarge van `0%` en een ondermarge van `5%` gekregen. Om het element te kunnen centreren moeten we de linker- en de rechtermarges als `auto` opgeven. Bijvoorbeeld:

```
body{
  margin-left: auto;
  margin-right: auto;
}
```

Border

Het `<body>`-element heeft een border (rand) van 1 pixel gekregen, als volgt:

```
border: 1px solid;
```

font

We gebruiken het attribuut `font` om het lettertype van de teksten binnen een element te definiëren. Hier volgt een lijst met lettertypes die geschikt zijn voor alle webbrowsers:

Figuur 1.19

font-size

Het attribuut `font-size` (tekengrootte) definieert de lettergrootte van je teksten. Dit attribuut kunnen we waarden geven in pixels, punten, percentages of relatief ten opzichte van het font in de body, bijvoorbeeld 1,5 keer de font-size van het `<body>`-element. Hieronder zien we hoe het font cerif gedefinieerd kan worden in pixels, punten, percentages of relatief ten opzichte van de basis:

`cerif 12px`	pixels
`cerif 24pt`	punten
`cerif 48%`	percentage
`cerif 1.5rem`	relatief

De meest exacte lettergrootte geven we aan in pixels (px).
Voor een betere kwaliteit afdruk van je teksten gebruik je punten (pt).
Percentages en rem zijn gerelateerd aan de lettergrootte van het `<body>`-element, bijvoorbeeld:

`body {font-size: 16px;}`	basis
`body h1 {font-size: 2.0rem;}`	relatief: 2.0 x 16 = 32px
`body h2 {font-size: 50%;}`	percentage: 50% van 16 = 8px

Subelementen

In onze webpagina zien we dat de `body` een `nav`-element heeft en dat het `nav`-element een `h3`-element heeft. Deze elementen noemen we subelementen. We kunnen heel specifiek een subelement selecteren door het pad van het element aan te geven. Bijvoorbeeld:

```
body nav {color: black}
```

of

```
body nav h3 {color: black}
```

font-weight

We kunnen ons lettertype verder vormgeven met `font-weight`. Hiervoor zijn er vier opties:

`normal`	normaal
`bold`	vet
`bolder`	vetter
`lighter`	lichter

box-shadow

Om de dimensies van onze elementen beter te zien hebben we in Webbouw 5 schaduwen gebruikt. In ons geval hebben we de volgende schaduw gedefinieerd:

```
box-shadow: 4px 4px 10px rgba(0,0,0,0.5);
```

Dit betekent:
– 4 pixels aan de rechterkant
– 4 pixels onderaan
– 10 pixels vervaging
– De kleur van de schaduw is zwart (0 rood, 0 groen, 0 blauw, dat is geen kleur, dus zwart) en is 50% transparant. Kleurencodes en de functie `rgba` bespreken we in de volgende paragraaf.

- *Vaardigheid-lab 02*
  In deze lab-opdracht codeer je stijlen voor de vier div-elementen zoals in de volgende figuur te zien is. Codeer een stylesheet met de volgende stijlen en sla die op als **labs.css**.

  Div 1 Border-shadow
  80% breed
  Hoogte 80px
  Font-size 20px
  Auto marge
  Marge boven 3%
  Marge onder 3%

  Div 2 80% breed
  Hoogte 220px
  Font-size 15px
  Linkermarge 5%
  Marge onder 3%

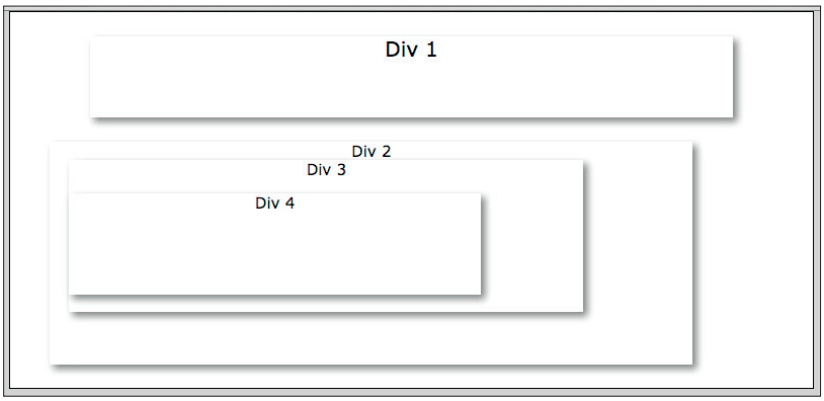
  Div 3 80% breed
  Hoogte 150px
  Linkermarge 3%

  Div 4 80% breed
  Hoogte 100px
  Linkermarge 0%

  Open **labs.html** en voeg de volgende link eraan toe:

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

  Het resultaat moet er als volgt uitzien:



Figuur Vaardigheid-lab 02

1.3 Webkleuren

In deze paragraaf maken we kennis met de volgende CSS-attributen:

- rgb
- rgba

Aan het einde van deze paragraaf laat je de volgende opgaven door je docent controleren:

Planning	Inleveren
	Oefening 7 t/m 9
	Vaardigheid-lab 03

Inleiding

Webkleuren zijn gebaseerd op de drie primaire kleuren: rood, groen en blauw. Door deze drie basiskleuren te combineren kunnen we een spectrum van kleuren creëren. In de volgende figuur zie je primaire, secundaire en tertiaire webkleuren.

Figuur 1.20 Kleurenschijf

Er zijn drie manieren om kleuren te produceren in CSS:

- met decimale getallen;
- met hexadecimale getallen;
- met percentages.

Omdat de browsers constant in ontwikkeling zijn en omdat de ene browser een betere manier vindt om bijvoorbeeld kleuren te coderen, ontstaan hierdoor meerdere manieren om kleuren te coderen.

De meest voorkomende manier om kleuren te coderen voor een webpagina is met hexadecimale getallen. Hexadecimaal is een numeriek systeem gebaseerd op tien getallen en zes letters (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F). We gaan niet dieper in op het hexadecimale systeem, maar het is belangrijk om te weten hoe het systeem wordt gebruikt om kleuren te combineren. Dit doen we met het attribuut `rgb`.

rgb (red, green, blue)

Met het attribuut `rgb` kunnen we de drie primaire kleuren, rood, groen en blauw mengen met als resultaat een nieuwe kleur. De syntaxis is als volgt:

```
rgb(red,green,blue)
```

De mogelijke waarden voor iedere kleur zijn de volgende:

- decimaal 0 t/m 255;
- percentage 0 t/m 100.

Bijvoorbeeld, de kleur blauw coderen we als volgt:

```
rgb(0,0,255)
```

Dit betekent: nul rood, nul groen en maximaal blauw.
We kunnen ook een beetje groen toevoegen, als volgt:

```
rgb(0,16,255)
```

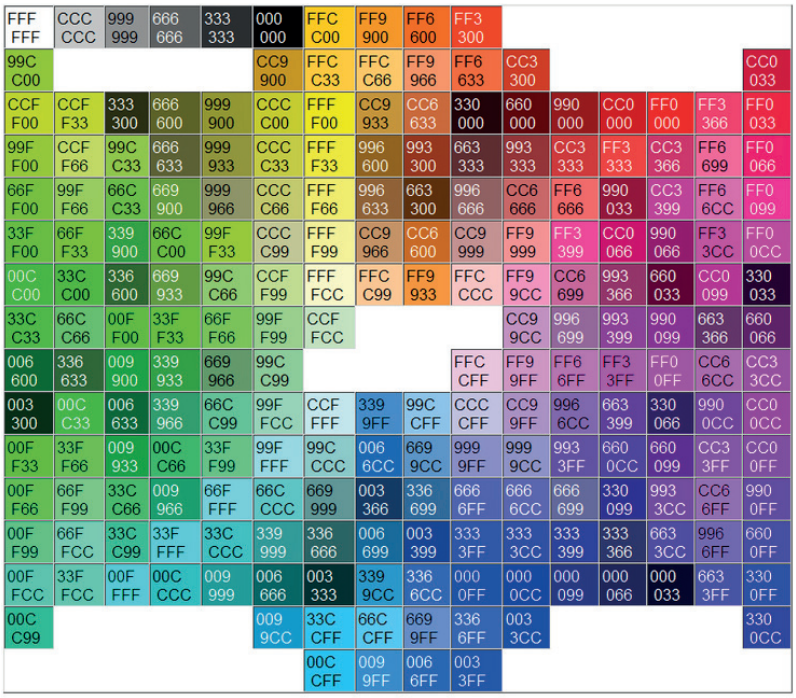
Dit geeft als resultaat een blauwgroene kleur. Blauw kunnen we in hexadecimale notatie als volgt coderen:

```
#0000FF
```

Hexadecimale notatie begint met een `#`-teken gevolgd door hexadecimale waarden voor rood, groen en blauw. In dit geval `00` rood, `00` groen en `FF` voor het maximale blauw. Als de twee getallen hetzelfde zijn, kunnen we op een verkorte manier coderen:

```
#00F
```

Op internet vind je talloze webpagina's met tabellen met kleurcodes zoals in de volgende figuur:



Figuur 1.21

Met de decimale methode kunnen we dezelfde groenblauwe kleur met de decimale waarden 0 t/m 255 als volgt creëren:

```
rgb(0,56,255)
```

Met percentages kunnen we dezelfde kleur als volgt creëren:

```
rgb(0%, 22%, 100%)
```

De afwezigheid van kleuren geeft als resultaat de kleur zwart:

```
rgb(0, 0, 0)
```

Hier zie je dat spaties na de komma optioneel zijn.

- *Oefening 7: background-color*

  Open een nieuw script en sla dit op als **oefening7.html**. Maak de volgende vijf div's met de aangegeven achtergrondkleuren. Voorzie de div's van een naam. Plaats tevens in de div's de kleur uitgeschreven. Zonder tekst is de div niet zichtbaar omdat deze geen hoogte heeft. Gebruik het attribuut `background-color`, bijvoorbeeld:

  ```
  background-color: #fff;
  ```

  Gebruik rgb of hexadecimale codes uit de volgende tabel.

kleur	percentage	decimaal	hexadecimaal	kort
zwart	rgb(0%,0%,0%)	rgb(0,0,0)	#000000	#000
rood	rgb(100%,0%,0%)	rgb(255,0,0)	#FF0000	#f00
groen	rgb(0%,0%,100%)	rgb(0,255,0)	#00FF00	#0f0
blauw	rgb(0%,0%,100%)	rgb(0,0,255)	#0000FF	#00f
wit	rgb(100%,100%,100%)	rgb(255,255,255)	#FFFFFF	#fff

Figuur 1.22

rgba

We hebben in Webbouw 5 al `rgba` gebruikt. We gebruiken het attribuut `rgba` om transparantie toe te voegen, waarbij `a` de transparantie aangeeft en een waarde tussen `0.0` en `1.0` krijgt. Bijvoorbeeld:

```
rgba(0, 255, 0, 0.3)
```

Hier krijgen we een groene kleur met een transparantie van `30%`. In de vorige opgave hebben we dit attribuut gebruikt om een zwarte schaduw met een transparantie van `50%` te creëren:

```
box-shadow: 4px 4px 10px rgba(0,0,0,0.5);
```

- *Oefening 8: rgba*

  Open een nieuw script en sla dit op als **oefening8.html**. Maak de volgende div's met de aangegeven transparanties in rood.

verzadigd rood
rgba(255,0,0, 1.0)
rgba(255,0,0, 0.9)
rgba(255,0,0, 0.8)
rgba(255,0,0, 0.7)
rgba(255,0,0, 0.6)
rgba(255,0,0, 0.5)
rgba(255,0,0, 0.4)
rgba(255,0,0, 0.3)
rgba(255,0,0, 0.2)
rgba(255,0,0, 0.1)
rgba(255,0,0, 0.0)

Figuur 1.23

- *Oefening 9: #kleurcodes*

  Open een nieuw script en sla het op als **oefening9.html**. Maak de volgende div's met de aangegeven kleurencodes:

#FE0000
#FF9C00
#F5FF00
#3CF10E
#01FFE5
#0024FF
#C500FF
#FF009C

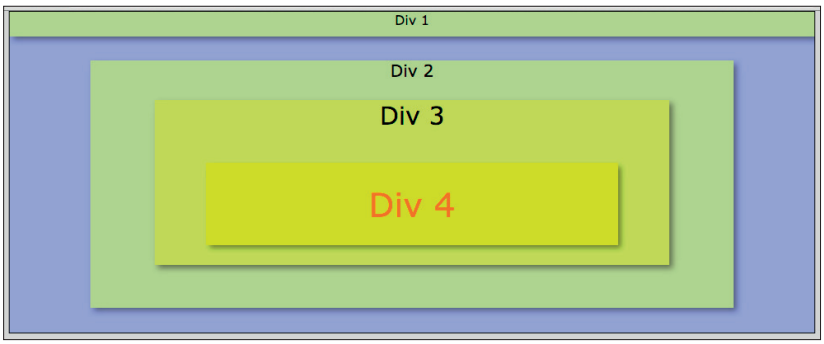
Figuur 1.24

- *Vaardigheid-lab 03*
  Open **labs.css** en geef iedere div een eigen achtergrondkleur, hoogte en breedte. Bij div 4 centreer je verticaal de tekst als volgt:

```
#div4 p{
  position: relative;
  top: 50%;
  transform: translateY(-50%);
}
```

`Position` wordt behandeld in paragraaf 1.5 en `transform` in paragraaf 1.6.

Het resultaat moet er als volgt uitzien:



Figuur Vaardigheid-lab 03

1.4 Navigatie

In deze paragraaf maken we de navigatie van de website en we maken kennis met een aantal HTML-elementen en een aantal CSS-attributen.

Aan het einde van deze paragraaf laat je de volgende opgaven door je docent controleren:

Planning	Inleveren
	Webbouw 7 t/m 11
	Vaardigheid-lab 04

Een goed gestructureerde navigatie bepaalt of je website logisch en intuïtief is. De navigatie moet zodanig gemaakt worden dat de gebruiker de gezochte informatie gemakkelijk kan vinden. We beginnen met de HTML-code en eindigen met de CSS-code.

Codering van de navigatie

De navigatie moet logisch opgebouwd worden zodat gezochte informatie nooit meer dan drie klikken verwijderd is van de homepage. Het moet mogelijk zijn om

vanuit iedere webpagina terug te keren naar de homepage. HTML5 heeft de nieuwe tag <nav> met als betekenis 'navigatie'.

Element <nav>

Meestal plaatsen we de navigatie bovenaan of aan de rechterkant van de pagina zodat we door de hele website makkelijk kunnen navigeren. Omdat meestal de navigatie in de vorm van een lijst wordt gecodeerd, kijken we nu naar HTML-lijsten.

Genummerde lijsten

Soms wil je een bepaald stuk tekst als lijst weergeven. Er bestaan twee soorten lijsten: genummerde lijsten en opsommingslijsten. We kijken eerst naar genummerde lijsten.

```
<ol>
  <li>Webdesign</li>
  <li>Webhosting</li>
  <li>Webprogrammering</li>
</ol>
```

Dit genereert het volgende resultaat: een genummerde lijst <ol> met drie items <li>:

1. Webdesign
2. Webhosting
3. Webprogrammering

Items

Alle lijsten moeten minimaal een item <li> hebben. Deze genummerde items worden automatisch genummerd volgens hun positie in de lijst.

Opsommingslijsten

Een opsommingslijst is een lijst met bullets. Bijvoorbeeld:

```
<ul>
  <li>Duurzaam</li>
  <li>Ecologisch</li>
  <li>Groene voetafdruk</li>
</ul>
```

genereert de volgende bulletlijst:

- Duurzaam
- Ecologisch
- Groene voetafdruk

Element <a>

Het element <a> is een klikbaar element voor het linken van twee documenten. Deze elementen noemen we hyperlinks. Een hyperlink coderen we als volgt:

```
<a href="index.html">Home</a>
```

Het <a>-element bestaat uit drie delen:

1. , met het attribuut href met de naam van het te linken document. In dit geval index.html.
2. De tekst waar we moeten klikken om te linken. In dit geval Home.
3. , om de hyperlink te sluiten.

Spaties

HTML bestaat voor Hyper Text Markup Language. De hypertekst is de content of de tekst tussen de tags in. In een hypertekst is het zo dat in de browser een, twee of meer spaties als één spatie worden gezien. Als we een extra spatie willen toevoegen gebruiken we de code . Bijvoorbeeld <h1> tekst</h1>.

  is de afkorting voor no-break space. Dit zorgt ervoor dat er een witruimte komt tussen twee woorden. Dit werkt anders dan een normale spatie omdat door   de woorden bij elkaar blijven. Als er geen ruimte is aan het eind van de regel gaan de twee woorden als één geheel naar de volgende regel.

In de volgende opgave vervangen we het <h3>-element binnen de navigatie <nav> door een lijst met hyperlinks.

```
<nav>
  <h3>Navigatie</h3>
</nav>
```

- ***Webbouw 7***

  Open **index.html** en vervang het <h3>-element binnen het <nav>-element door de volgende lijst met hyperlinks.

```
<nav>
    <ul>
      <li><a href="index.html">Home</a></li>
      <li><a href="playlist.html">Playlist</a></li>
      <li><a href="formulier.html">Formulier</a></li>
      <li><a href="contact.html">Contact</a></li>
    </ul>
  </nav>
```

De volgende figuur illustreert het resultaat van de opgave: de navigatie als een bulletlijst met hyperlinks weergegeven.

- Home
- Playlist
- Formulier
- Contact

LET OP

Als je een ander resultaat krijgt, ga dan terug naar je code en je stijlen en zoek uit wat er fout is gegaan.

Figuur 1.25 Resultaat van Webbouw 7

Ontwerp van de navigatie

Hier gaan we verder met het ontwerp van de navigatie van onze website. In dit deel kijken we naar de volgende attributen:

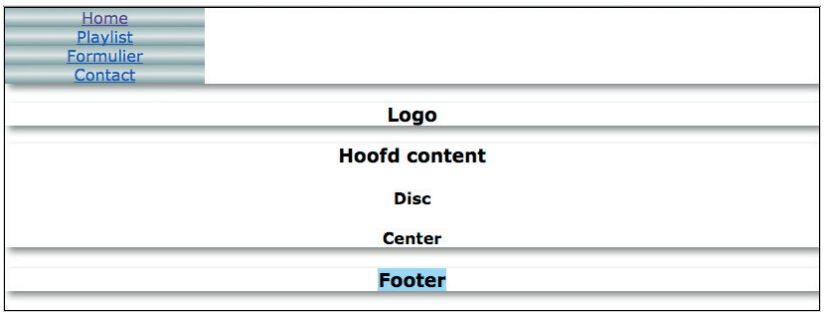
- list-style-type
- display: block
- float
- gradient
- tekst-decoration
- opacity
- pseudoklassen

- *Webbouw 8*

  Open **styles.css** en voeg de volgende stijlen eraan toe:

```
nav {
  width: 100%;
}
nav ul {
  text-align: center;
  margin: 0 0 0 0;
  padding-left: 0;
}
nav ul li {
  list-style-type: none;
}
nav ul li {
  width: 25%;
  background: linear-gradient(#9FB3B5, #E3E9EA, #7C9799);
}
```

Het resultaat ziet er als volgt uit:



Figuur 1.26

Hier volgt een uitleg over Webbouw 8.

list-style-type

Dit attribuut is voor de opmaak van lijsten. We willen onze hyperlinks hebben in een lijst zonder bullets. Dit doe je als volgt:

```
nav ul li {
  list-style-type: none;
}
```

Naast de optie `none` kunnen we de volgende opties gebruiken:

`disc`	disc	●
`circle`	cirkel	○
`square`	vierkant	□
`decimal`	decimaal	1
`lower-latin`	kleine letter Latijn	i
`upper-latin`	hoofdletter Latijn	I

Het `<body>`-element bevat het `<nav>`-element en dat krijgt als breedte `100%` van het `<body>`-element. In de vorige opgave hebben we een navigatie gemaakt met vier hyperlinks, dus krijgt iedere link `25%` van de breedte van de navigatie.

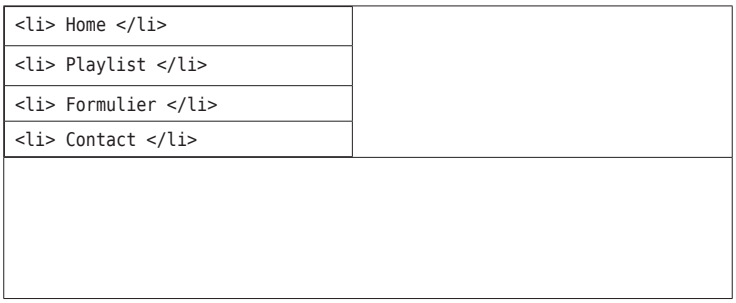
```
nav ul li {
  width: 25%;
}
```

display: block

De volgende HTML-elementen krijgen automatisch het attribuut `display: block`.

- `<h1>` t/m `<h6>`
- `<p>`
- `<li>`
- `<div>`
- `<main>`
- `<footer>`
- `<section>`

Onze `<li>`-elementen binnen de navigatie krijgen ieder `25%` van de breedte van het nav-element en worden als blokken onder elkaar weergegeven.



float

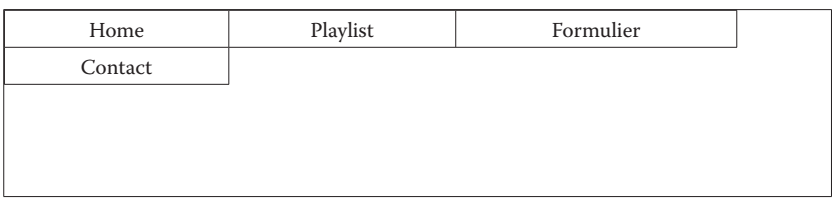
Om onze hyperlinks horizontaal te groeperen gebruiken we het attribuut `float`. Elementen met dit attribuut hebben de mogelijkheid om naar links of rechts te 'zweven'.

`float: left` naar links zweven
`float: right` naar rechts zweven

Neem als voorbeeld de letters in een tekst: alle letters hebben de neiging om naar links te 'zweven' naar de vorige letter. Bijvoorbeeld:

Maar als het scherm te klein is gaat de letter onderaan naar de volgende regel, bijvoorbeeld:

Dat gebeurt ook met HTML-elementen met het attribuut `float`. Als we bijvoorbeeld onze `<li>`-elementen een breedte van `30%` geven dan 'zweeft' het vierde element onderaan naar de volgende beschikbare regel.



clear

Met het attribuut `clear` kunnen we het 'zweven' ongedaan maken.

`clear: left` naar links zweven ongedaan maken
`clear: right` naar rechts zweven ongedaan maken
`clear: both` naar links en rechts zweven ongedaan maken

gradient (kleurverzadiging of kleurverloop)

Als laatste hebben we in de vorige opgave het attribuut `linear` kleurverloop gecodeerd. Met dit attribuut vullen we een achtergrond in met een kleurverloop. Er zijn twee soorten `gradients`: `linear` en `radial`. De syntaxis is als volgt:

```
linear-gradient(kleur-1, kleur-2, ...);
```

Bijvoorbeeld:

```
linear-gradient(red, blue);
linear-gradient(red, yellow, blue);
```

De volgende figuur illustreert het resultaat van de tweede regel: een lineair kleurverloop van rood naar geel naar blauw.

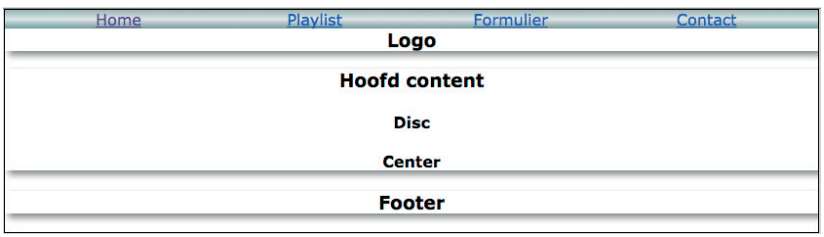
Figuur 1.27

We kunnen ook een diagonaal kleurverloop coderen:

```
linear-gradient(45deg, red, yellow, blue);
linear-gradient(to left top, red, yellow, blue);
linear-gradient(to bottom, red, yellow, blue);
```

- ***Webbouw 9***

Open **styles.css**. In deze opgave selecteer je het `<li>`-element binnen onze navigatie en past het attribuut `float left` toe. Het resultaat moet er als volgt uitzien:



Figuur 1.28

In de volgende opgave gaan we verder met de navigatie. We kijken naar meer attributen voor de hyperlinks.

- ***Webbouw 10***
  Open **styles.css** en voeg de volgende stijlen eraan toe:

```
a {
  text-decoration: none;
}

nav ul li a {
  font-size: large;
  opacity: 0.5;
  filter: alpha(opacity=50);
}
nav ul li a:hover,
nav ul li a:focus {
  color: firebrick;
}
```

Hier krijgen de hyperlinks een lettergrootte van `1.7` in relatie tot de `16px` basislettergrootte van het `body`-element. Omdat `hover` en `focus` geen attributen maar effecten zijn, plaatsen we geen spaties na de dubbelepunt, dus we schrijven: `a:hover` en `a:focus`.

text-decoration

Met het attribuut `text-decoration` kunnen we de volgende opties toepassen: doorstrepen, onderstrepen, bovenstrepen.

`none`	geen
`underline`	onderstrepen
`overline`	lijn erboven
`line-through`	~~doorhalen~~

Alle hyperlinks krijgen automatisch een onderstreping. Onderstrepen schakelen we als volgt uit:

```
text-decoration: none;
```

opacity

Het attribuut `opacity` bepaalt de transparantie van een element. In de vorige opgave kregen alle hyperlinks een transparantie van 50%.

```
opacity: 0.5;
```

pseudoklassen

Het beeldscherm is de interface van de browser. Een interface is het punt van interactie tussen twee systemen of objecten. Browsers hebben listeners (luisteraars) die permanent aan het luisteren zijn naar events (gebeurtenissen) op het beeldscherm.

Bijvoorbeeld wanneer de muis over een tekst zweeft. Dit noemen we muis-events. Muis-events kunnen we programmeren met behulp van pseudoklassen. Hyperlinks hebben de volgende pseudoklassen:

`a:hover`	wanneer de muis over een hyperlink zweeft
`a:focus`	wanneer de muis op een hyperlink focust
`a:visited`	wanneer de muis een hyperlink heeft geactiveerd
`a:active`	wanneer de muis een hyperlink activeert

De pseudoklassen `a:visited` en `a:active` veranderen de kleur van de bezochte en de actieve hyperlinks. Zo kunnen we zien welke links we bezocht hebben. In de laatste opgave hebben we met behulp van de pseudoklasse `a:hover` een hover-effect gecodeerd zodat de kleur van de tekst van alle hyperlinks verandert wanneer de muis over de link zweeft.

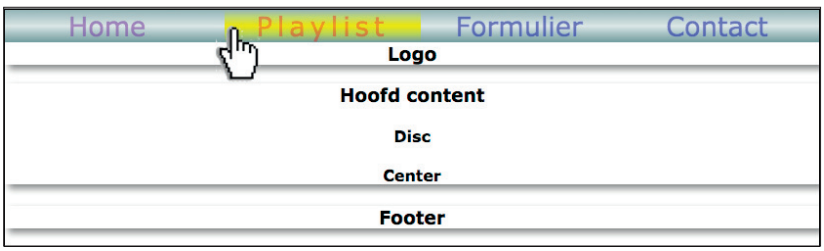
- *Webbouw 11*

  Om de navigatie makkelijker te maken voor onze gebruikers codeer je in deze opgave een hover-effect zodat de achtergrondkleur van de hyperlinks verandert wanneer een hover-event plaatsvindt.

  Daarna codeer je een tweede hover-effect voor kleurenblinde gebruikers waardoor het hover-event de ruimte tussen de letters van een hyperlink laat veranderen. Gebruik hiervoor de volgende code:

```
letter-spacing: 5px;
```

  Het resultaat moet zoiets zijn:

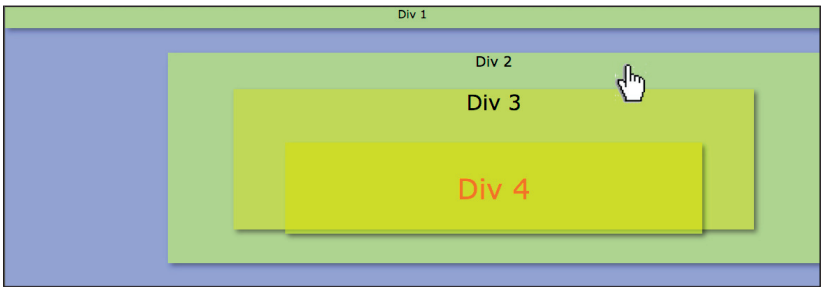


Figuur 1.29

Je ziet dat bij het hover-effect de ruimte tussen de letters verandert.

- *Vaardigheid-lab 04*

  Open **labs.css**. In deze lab-opdracht codeer je een hover-effect zodat div 2 naar rechts zweeft met een muis-over-event. Het resultaat moet er als volgt uitzien:



Figuur Vaardigheid-lab 04

1.5 Flexbox

Flexbox is een CSS-module die zorgt voor de verdeling en positionering van de HTML-elementen in kolommen of rijen. Daardoor hoeven we de percentages, paddings en marges van elementen niet te specificeren en uit te rekenen. Met flexbox declareren we CSS-properties voor een flex-container en voor de items in de container.

Planning	Inleveren
	Oefeningen 10 t/m 18
	Webbouw 12

- *Oefening 10: Flex-box*

  Open een nieuw script en sla dit op als **flexbox.html.** Typ de volgende code.

```
<!DOCTYPE html>
<html lang="nl">

<head>
  <meta http-equiv="Content-Type" content="text/html;
    charset=UTF-8">
  <meta name="robots" content="all">
  <link rel="stylesheet" href="flexbox.css">
<title>Flexbox</title>
</head>
<body>
  <container>
    <div class="a">A</div>
    <div class="b">B</div>
    <div class="c">C</div>
  </container>
</body>
</html>
```

Hier zijn `div`'s `a`, `b` en `c` de items in de container.

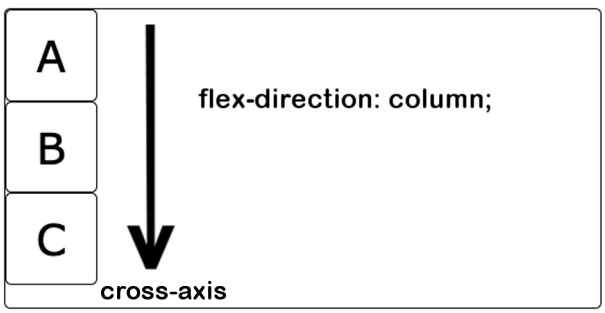
- *Oefening 11: Flex-box-properties*
  Open een nieuw script en sla dit op als **flexbox.css**. Typ de volgende code.

```
container {
  border: solid 1px;
  border-radius: 5px;
  width: 500px;
  height: 250px;

  display: flex;
  flex-direction: column;
}

div {
  border: solid 1px;
  border-radius: 5px;
  height: 75px;
  line-height: 75px;
  width: 75px;
  font-family: Verdana, Arial, sans-serif;
  color: #000;
  font-size: 36px;
  text-align: center;
}
```

Container-properties

Flex-direction

Hier hebben we de items van de container in de richting van de cross-axis in een kolom geplaatst. Het resultaat zie je hieronder.



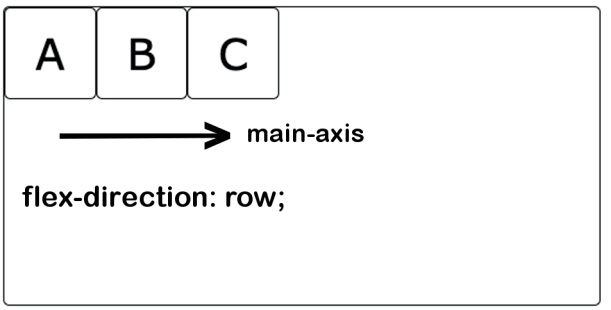
Figuur 1.30

We kunnen de items van een container als een rij plaatsen. Dit noemen we de richting van de main-axis.

- *Oefening 12: Flex-direction*
  Open **flexbox.css** en voeg de volgende property aan de container toe:

```
flex-direction: row;
```

Hier hebben we de items langs de main-axis geplaatst.



Figuur 1.31

We kunnen ook de volgende richtingen gebruiken:

```
flex-direction: row-reverse;
```

en

```
flex-direction: column-reverse;
```

Align-items langs de cross-axis

Om de items van een container uit te lijnen langs de cross-axis gebruiken we de property `align-items`. We kunnen een van de volgende opties gebruiken:

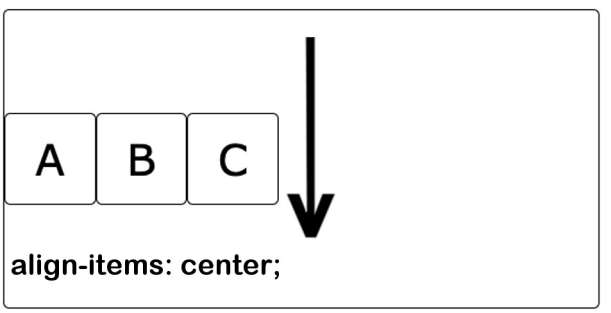
```
align-items: flex-start;
align-items: center;
align-items: flex-end;
```

- *Oefening 13: align-items*

  Open **flexbox.css** en voeg de volgende property aan de container toe:

```
align-items: center;
```

Hier hebben we de items langs de cross-axis gecentreerd.



Figuur 1.32

Justify-content langs de main-axis

Om de items langs de main-axis uit te lijnen gebruiken we `justify-content`. We kunnen een van de volgende opties gebruiken.

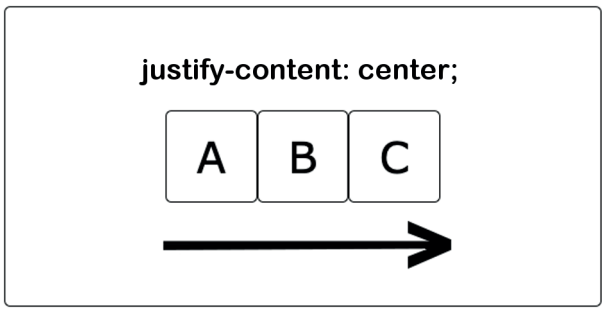
```
justify-content: flex-start;
justify-content: center;
justify-content: flex-end;
```

- *Oefening 14: justify-content*

Open **flexbox.css** en voeg de volgende property aan de container toe:

```
justify-content: center;
```

Hier hebben we de items langs de main-axis gecentreerd.



Figuur 1.33

Ruimteverdeling

Als er ruimte langs de x- of cross-axis over is kunnen we de items als volgt indelen:

```
justify-content: space-between;
justify-content: space-evenly;
justify-content: space-around;
```

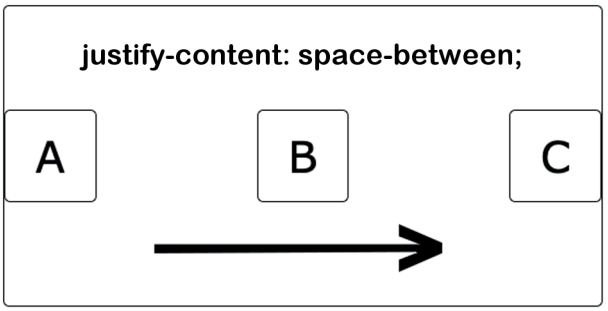
De property `justify-content` kunnen we in rows of in columns gebruiken.

- *Oefening 15: Ruimte indelen*

Open **flexbox.css** en voeg de volgende property aan de container toe:

```
justify-content: space-between;
```

Hier hebben we de ruimte tussen de elementen langs de main-axis ingedeeld.

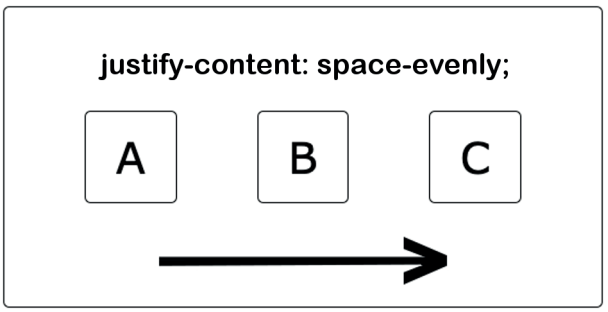


Figuur 1.34

- *Oefening 16: Ruimte gelijkmatig indelen*
  Open **flexbox.css** en voeg de volgende property aan de container toe:

```
justify-content: space-evenly;
```

Hier hebben we de ruimte tussen de elementen gelijkmatig langs de main-axis ingedeeld.



Figuur 1.35

Item-properties

We kunnen properties aan individuele items toewijzen.

align-self

De property `align-self` van een item overschrijft de property `align-item` in de container. Bijvoorbeeld:

```
align-self: flex-end;
```

order

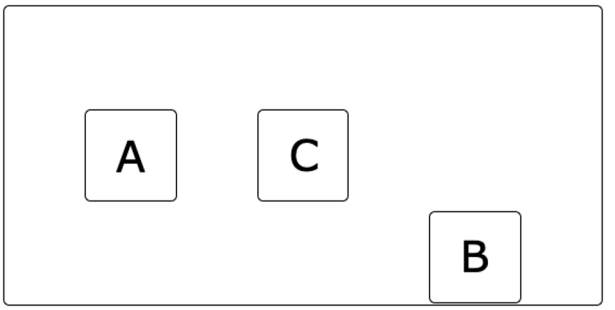
De property `order` van een item overschrijft de volgorde van de gecodeerde items in het HTML-script. De syntaxis is:

```
.item {
  order: <integer>; /* default is 0 */
}
```

- *Oefening 17: item orde*
  Open **flexbox.css** en voeg de volgende properties aan item B toe:

```
order: 3;
align-self: flex-end;
```

Hier hebben we het tweede item B als laatste vertoond en onderaan uitgelijnd.



Figuur 1.36

Flex items

We kunnen de vrije ruimte per individueel item indelen door een van de volgende opties te kiezen:

```
flex-grow: initial;
flex-shrink: initial;
flex-basis: 40%;
```

- *Oefening 18: flex items*
  Open **flexbox.css** en voeg de volgende properties aan items A, B en C toe:

```
.a {
  flex-grow: 2;
}

.b {
  flex-basis: initial;
}

.c {
  flex-basis: initial;
}
```

Hier hebben we items B en C met hun aanvankelijke breedte van 75 px weergegeven en item A neemt de resterende 350 px vrije ruimte in.