

Canvas en SVG – Animaties en vectoren

Joris Geens



Download free books at

bookboon.com

Joris Geens

Canvas en SVG – Animaties en vectoren

Canvas en SVG – Animaties en vectoren

1ste editie

© 2015 Joris Geens & bookboon.com

ISBN 978-87-403-0976-8

Peer review door Sven Maes, Leerkracht, Onderwijs Vlaanderen

Inhoud

Soopbooks	10
Voorwoord	11
Deel 1 – Keuze technologie	13
1 Canvas of svg?	14
1.1 Illustraties	14
1.2 Animaties	14
1.3 Canvas of svg?	14
2 Indeling boek	15
3 Voorkennis	16
4 Objecten	17
5 Oefeningen	18

We ask you
**WHERE DO YOU
 WANT TO BE?**

TOMTOM

TomTom is a place for people who see solutions when faced with problems, who have the energy to drive our technology, innovation, growth along with goal achievement. We make it easy for people to make smarter decisions to keep moving towards their goals. If you share our passion - this could be the place for you.

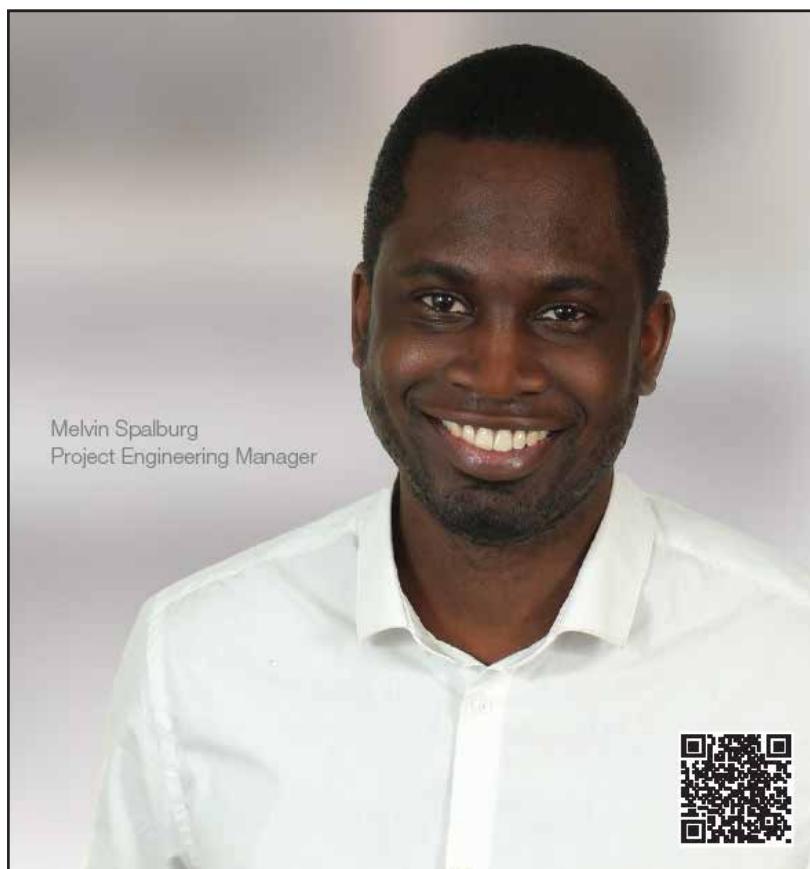
Founded in 1991 and headquartered in Amsterdam, we have 3,600 employees worldwide and sell our products in over 35 countries.

For further information, please visit tomtom.jobs

Download free eBooks at bookboon.com



Deel 2 – Canvas	19
1 Doelen	20
2 Inleiding	22
2.1 Grootte bestanden	22
2.2 Bewerking en interactiviteit	22
2.3 Rasterafbeeldingen	22
3 Canvas-container	23
3.1 Html5-structuur	23
3.2 Canvas in body	24
3.3 Identiteit	25
3.4 Opmaak	26
3.5 Samenvatting	27
4 Context	28
4.1 Gereedschapskist	28
4.2 Bewaren en ophalen	32



Melvin Spalburg
Project Engineering Manager

AkzoNobel

Improve
that plant's
PERFORMANCE
together

To find out more about Melvin,
please visit our website
www.akzonobel.nl/careers

07251_190813

5	Paden	34
5.1	Begrip	34
5.2	beginPath()	34
5.3	closePath()	37
6	Teksten	38
6.1	Lettertype en grootte	38
6.2	Schrijven	40
6.3	Kleur	42
6.4	Uitlijning	44
6.5	Basislijn	45
7	Vullen en omlijnen	47
7.1	Vullen	47
7.2	Omlijnen	57
8	Lijnen	59
8.1	Rechte lijn	59
8.2	Kromme	64
9	Joins van lijnen	71



**WERKEN
3.0?**

EARTH, LIFE AND SOCIAL SCIENCES

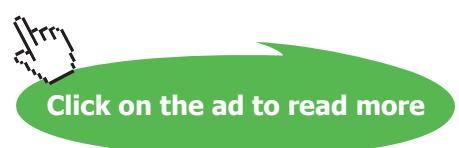
INNOVATIES DIE WERKEN

Mensen, organisaties en onze samenleving bepalen uiteindelijk welke innovaties voor hen van groot belang zijn en op welke wijze ze daarmee kunnen en willen werken. Daarvoor heb je innovaties nodig, maar ook veel kennis van menselijk gedrag - zowel individueel als in teams - van organisaties, maar ook van de stuwende kracht achter maatschappelijke processen. Die kennis levert het expertisegebied Earth, Life and Social Sciences.

Zo'n 1300 mensen variërend van onder meer moleculair bioloog, chemicus, ICT'er, bedrijfskundige, wiskundige, psycholoog en aardwetenschapper zijn hierbij betrokken. Juist de synergie van creatieve combinaties van vakgebieden maakt het ons mogelijk om uitdagende problemen voor onze klanten aan te pakken en helpen op te lossen.

TNO innovation for life

TNO.NL/CAREER



10	Rechthoeken	73
10.1	Rechthoek	73
10.2	Varianten	75
10.3	Uitsnijden	77
11	Cirkels	79
11.1	Volle cirkel	79
11.2	Deel cirkel	80
12	Clipping	82
12.1	Bewaren	84
12.2	Gastheer	84
12.3	Clip-figuren	85
12.4	Terugkeren	85
13	Transformaties	86
13.1	Schalen	86
13.2	Roteren	88
13.3	Vervormen	89
13.4	Verplaatsen	91
13.5	Reset	92

**DENMARK
IS HIRING**



Are you looking to further your cleantech career
in an innovative environment with excellent
work/life balance? Think Denmark!
Visit cleantech.talentattractiondenmark.com

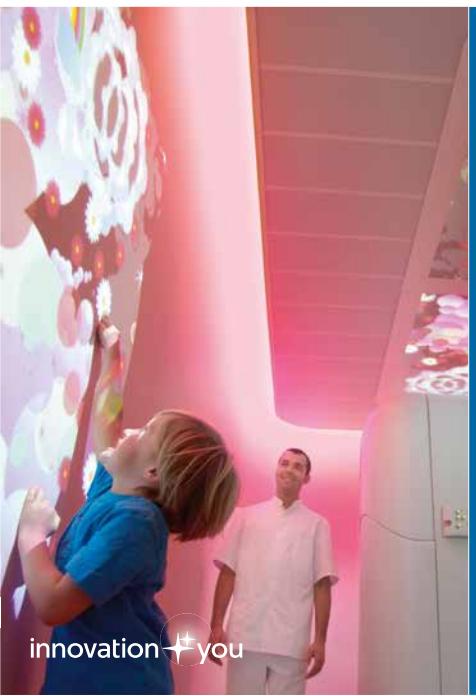
*"In Denmark you can find great engineering jobs and develop
yourself professionally. Especially in the wind sector you can learn
from the best people in the industry and advance your career in a
stable job market."*

Mireia Marrè,
 Advanced Engineer from Spain.
 Working in the wind industry in Denmark since 2010.

**Empower
your career**
- think Denmark



14	Afbeeldingen	96
14.1	Laden en tonen	96
14.2	Grootte	98
14.3	Uitsnijden	99
15	Effecten	102
15.1	Schaduw	102
15.2	Transparantie	104
16	Opslaan als afbeeldingbestand	106
16.1	Code	106
16.2	Browser	108
17	Animatie	109
17.1	Frame	109
17.2	JavaScript	111
17.3	requestAnimationFrame()	112
17.4	setInterval()	113



Some see lighting that captivates. **You see hospitals that comfort**

Imagine the impact you can have
 Philips is a diversified technology company, focused on improving people's lives through meaningful innovations. As a world leader in Healthcare, Consumer Lifestyle and Lighting, Philips integrates technologies and design into people-centric solutions, based on fundamental customer insights.

Meaningful contribution
 At Philips, we strive to make the world healthier and more sustainable through innovation. Our goal is to improve the lives of 3 billion people a year by 2025. To achieve this we are looking for passionate, insightful people that have the drive to deliver meaningful results together with a great team.

If you want to deliver a meaningful contribution and want to have an impact on the quality of people's lives, take your chance to start your future at Philips as an intern, a trainee or as a talent in a regular job.

Find out more on www.philips.nl/careers

PHILIPS



Click on the ad to read more

18	Oefenen	115
18.1	Checklist	115
Deel 3 – Svg		116
1	Doelen	117
2	Inleiding	118
2.1	Vectoren	118
2.2	Inkscape	119
2.3	SVG-dom	120
3	<svg>-container	121
4	Vormen	123
4.1	Lijn	123
4.2	Rechthoek	124
4.3	Cirkel	126
4.4	Ellips	128
4.5	Veelhoek	129
4.6	Veelvoudige lijn	130
5	Paden	134
5.1	move to	134
5.2	line to	135
5.3	Andere	136
6	Teksten	137
6.1	Tekst tekenen	137
6.2	Tekst indelen	138
7	Animatie	139
7.1	Transparantie	139
7.2	Vervormen	141
8	Oefenen	143
8.1	Checklist	143

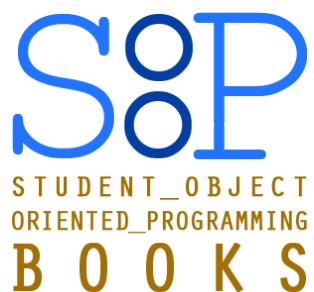
Soopbooks

Alle voorbeelden, uitgewerkte bronbestanden, oefeningen en oplossingen staan op www.soopbooks.com voor je klaar.

Soopbooks staat voor *Student Object Oriented Programming Books*. De boeken richten zich op iedereen die graag wil leren coderen, scripten en programmeren. Je ontdekt een boeiende wereld, zonder saaie theorieën, vol leuke voorbeelden.

Kom nu al meteen even langs en schrijf je alvast op de nieuwsbrief in!

De auteur is van het menselijke ras. Daardoor valt het niet uit te sluiten dat er hier of daar een foutje in het boek geslopen is. Denk je een fout gevonden te hebben, of heb je een suggestie, laat het de auteur dan weten op goedfout@soopbooks.com. Vergeet niet om de titel van dit boek en het bijbehorende paginanummer te vermelden. De auteur is je eeuwig dankbaar!



Voorwoord

Wie een website bouwt, kan niet zonder illustraties of animaties. Een webstek zonder icoontjes of bewegende figuurtjes is hopeloos verouderd.

Vroeger tekende je zulke illustraties met tekensoftware en bouwde je animaties met Flash. Die tekenpakketten doen vandaag nog steeds hun ding, toch komt html nu als een waardig alternatief opdraven. Wat animaties betreft, hoef je je niet langer meer te buigen over de verouderde Flash-technologie. Vandaag bouw je animaties met technologieën die je sowieso al nodig hebt bij webontwerp, namelijk html, css, Javascript en xml.

Hoe leren?

Leer je illustreren en animeren met de vingers in de neus? Ja en nee. Wanneer je dit boek doorwerkt, zal je ontdekken hoe eenvoudig html, css, JavaScript en xml wel kunnen zijn. Toch volstaat het niet om het boek simpelweg te lezen. Probeer de voorbeelden actief uit, maak oefeningen en pas je verworven kennis meteen ook op je eigen project toe. En vooral, webdevelopment is niet alleen het leren van codes en regeltjes, het blijft ook een vaardigheid, die je alleen door veel te doen onder de knie krijgt. Doen, doen, doen, is dus de boodschap.

Loop je doorheen de hoofdstukken ergens vast? Raak je uit bepaalde stukken code echt niet uit? Stel je vraag dan gerust op het forum bij dit boek, dat je op www.soopbooks.com ontdekt. Je bent trouwens altijd van harte welkom op die website, daar vind je alle bestanden en oefeningen die bij dit boek horen.

Dank

Tot slot van dit voorwoord neem ik graag ook even de tijd om met oprechte dankbetuigingen te zwaaien.

Op de eerste plaats dank ik mevrouw Karin Hamilton Jakobsen, Managing Editor bij Bookboon.com, voor een bijzonder puike samenwerking. Ook dank ik de heren Kristian Buus Madsen en Thomas Buus Madsen hartelijk, respectievelijk ceo en coo van Bookboon.com, voor het opnemen van dit boek in het enorme aanbod van Bookboon.com.

Vervolgens, dank aan de heer Sven Maes, die het boek niet alleen inhoudelijk doorploegde, maar het ook kritisch didactisch en taalkundig nakeek.

Als voorlaatste in de rij wil ik graag ook mijn goede vriend www.w3c.org in de verf zetten. Die website stond voor mij altijd klaar – dag en nacht, zowel op werkdagen als in het weekend – tijdens de uitwerking van dit boek.

Tot slot dank ik ook de lezer. Bedankt om dit boek *Canvas en svg – Animaties en vectoren* te downloaden! Enneuh, bezorg me gerust een link naar jouw eigen trotse canvas- of svg-project op www.soopbooks.com, dan voeg ik die graag aan de eregalerij toe. Veel plezier met dit boek!

Joris Geens

auteur

Deel 1 – Keuze technologie

1 Canvas of svg?

Welkom in de webwereld van de illustratie en de animatie! Leer met de html5-canvas, css en JavaScript, of svg en xml knappe illustraties te ontwerpen.

1.1 Illustraties

Enkele jaren geleden was het ondenkbaar om met html een tekening op een webpagina te maken. Je schetste de illustratie toen in een tekenpakket en je bewaarde die bijvoorbeeld als een gif of een jpg. Met de ``-tag toverde je zo'n jpg-afbeelding dan zo op het webscherf. Een overbodig geworden werkijze? Nee, toch niet. Maar er bestaan tegenwoordig wel waardevolle alternatieven.

Vandaag kan je illustraties namelijk op basis van zuivere code op het scherm brengen. Meer zelfs, met html en JavaScript, of met html, svg en xml. En dat heeft zo zijn voordelen. Denk maar aan de snelheid waarmee lijnen code van de webserver over het internet naar de browser bij de client flitsen, in tegenstelling tot tragere afbeeldingbestanden.

Hoe fiks je dat in een gewoon html-bestand? Wel, met de `<canvas>`-container kan je zulke illustratiecode binnen je html onderdak bieden. Voeg aan die container nog JavaScript toe en je bent bitmapsgewijs vertrokken. Of voeg aan je html een `<svg>`-container toe en volg de vectoriële weg.

1.2 Animaties

Wilde je een paar jaar geleden animaties voor het web ontwerpen, dan koos je vaak voor Flash. Echter is Flash vandaag helemaal voorbij gestoken door het drieluik html5, css en JavaScript. Maak je echter niet ongerust. Niet dat je alles van nul af aan zal moeten coderen, er bestaan heel wat goede programma's die je daar visueel bij helpen. Dat kan bijvoorbeeld met Adobe Edge of Google Webdesigner.

1.3 Canvas of svg?

Kies je voor de eenvoudigere canvas of voor het gevorderde svg? Laten we met de deur in huis vallen. Een belangrijk nadeel van die html-canvas heet bitmap. Een canvas-illustratie vormt namelijk een rasterafbeelding. Elke illustratie wordt dan als een verzameling van losstaande pixels beschouwd. Zoom je op zo'n rasterafbeelding fel in, dan zal je die pixels letterlijk zien verschijnen.

Vormt dat een obstakel voor jou? Kies dan voor het vectoriële svg-formaat dat ook volledig aan de hand van codes op te bouwen is. Op een dergelijke afbeelding mag je blijven inzoomen, de kwaliteit van de lijnen blijft steeds haarscherp, vandaar ook de naam *scalable vector graphics*.

Svg hoeft geen directe samenwerking met JavaScript. Een svg-bestand bestaat namelijk uit zuivere xml, die een browser probleemloos in een keurige afbeelding kan vertalen. Elke moderne browser heeft immers standaard een xml-parser aan boord.

2 Indeling boek

Dit boek bestaat uit drie delen. In het eerste deel – dat je nu leest – leggen we het verschil tussen de canvas en svg uit, ontdek je de indeling van dit boek en bekijken we de voorkennis die je nodig hebt om met dit ebook aan de slag te kunnen gaan. Vervolgens gaan we nog even wat dieper op objecten in en vertellen we jou hoe je op deze materie kan oefenen.

I WANT
CHALLENGES
THAT DEMAND
INNOVATIVE
SOLUTIONS

MY TIME IS NOW.

Coca-Cola Enterprises
Thirst.

Download free eBooks at bookboon.com



Click on the ad to read more

3 Voorkennis

Om dit boek vlot te kunnen gebruiken, bezit je best een basiskennis van html5 en JavaScript. Heb je die nog niet in huis, lees dan ook het boek HTML5 en CSS3 en het boek JavaScript en JQuery van deze auteur, uitgegeven bij Bookboon.com.

Daarnaast is het handig dat je ook een basiskennis van xml bezit.

Soopbooks.com

Kijk ook op www.soopbooks.com. Regelmatig verschijnen er nieuwe titels over andere onderwerpen.

4 Objecten

Wanneer je met de html5-canvas of met svg aan de slag gaat, kan je niet om het concept object heen.

De term object komt uit de wereld van het object georiënteerd programmeren, kortweg oop. Programmeertalen zoals Java, VB.net, C, C# of Objective-C steunen allemaal op dat oop-principe. Dat betekent dat de talen met classes – blauwdrukken met code – werken.

In een dergelijke class stoppen de makers ervan welbepaalde functionaliteiten. Dat kan bijvoorbeeld code zijn die ervoor zorgt dat tekst op het scherm verschijnt. Echter kan een programmeur nooit rechtstreeks met een class aan de slag, hij maakt er steeds een kopietje van. Zo staat de oorspronkelijke code veilig en blijft ze onaangeroerd. Zo'n kopie noemen we een object.

JavaScript is ook een object georiënteerde taal. Echter is het een bijzondere oop-taal. JavaScript is namelijk een prototype-oop-taal. Dat betekent dat die programmeertaal weliswaar met objecten werkt, maar dan zonder classes, een belangrijk verschil met de hogere programmeertalen. Al het codebronmateriaal bevindt zich dan in objecten in plaats van classes. Die bronobjecten mag je evenmin rechtstreeks gebruiken, ook daar moet je een kopietje van nemen. Zo'n kopie noemen we eveneens een object.

Wat zit er nu precies in een object? Een object is een verzameling van eigenschappen en methodes rond een bepaald thema. In de mensenwereld zou een object bijvoorbeeld een smartphone kunnen zijn. Zo'n telefoon kent eigenschappen. Denk maar aan een kleur, een gewicht, materiaal, afmetingen, ... Daarnaast bezit zo'n smartphone ook methodes, ofwel de dingen die je ermee kan doen. Zo kan je met een smartphone bellen, sms'en, internetten, ... In de digitale wereld is dat niet anders. Zo kan het JavaScript-object `document` de eigenschap `title` bevatten, alsook de methode `write()`.

Objecten

Wil je meer over objecten en JavaScript weten? Kijk dan in het boek Javascript en JQuery van dezelfde auteur.

5 Oefeningen

Wil je oefenen op de canvas of op svg? Kijk dan op claimjekennis.soopbooks.com. Daar ontdek je oefeningen.

Start jij jouw carrière als Junior Product Developer
Innovatieve Printsystemen?

(Bijna) klaar met je WO-opleiding? Start bij Océ als Product Developer en werk aan
inkten, printheads en print processen van de toekomst! Interesse? **Lees hier meer!**

océ
A CANON COMPANY



Deel 2 – Canvas

1 Doelen

Als je dit tweede deel doorgenomen hebt, bereik je de onderstaande doelstellingen. Je kan op dat moment je kennis en vaardigheden ook even testen met de oefeningen aan het einde van dit deel.

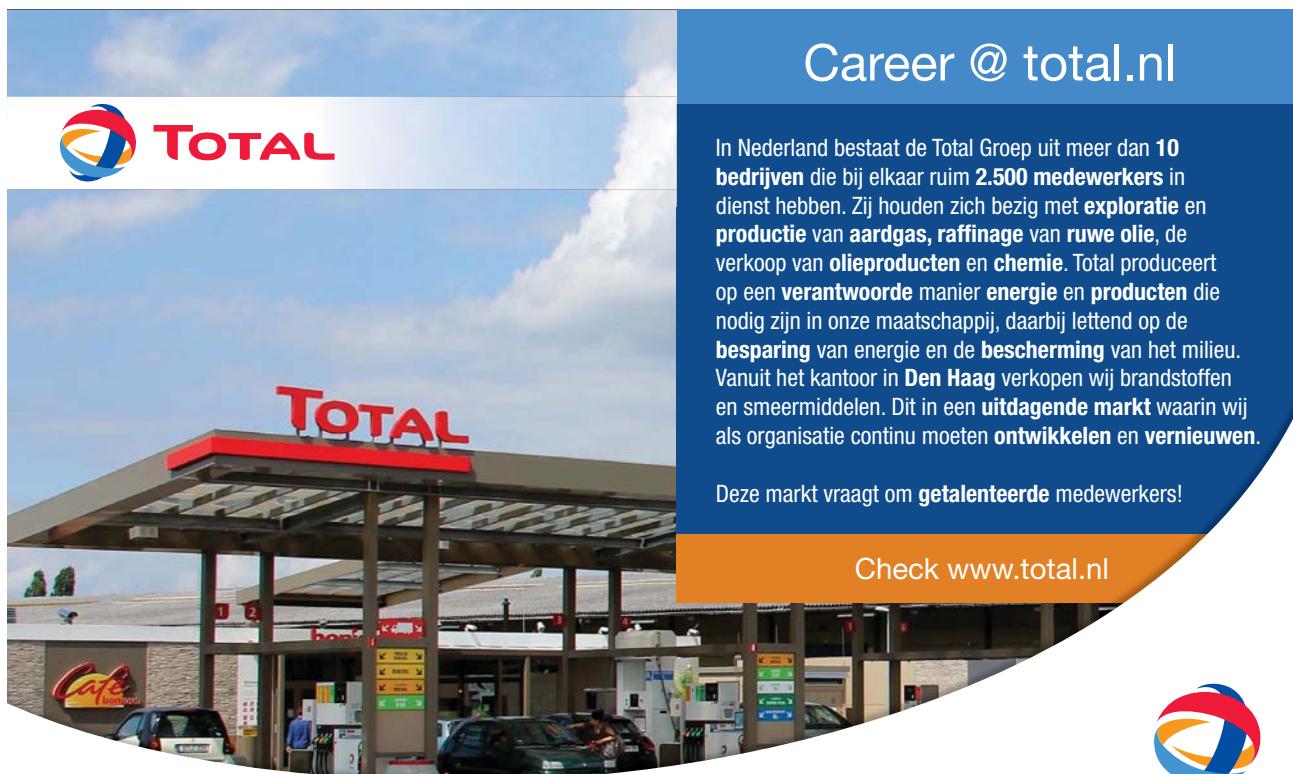
Overigens vormen die oefeningen een absolute aanrader, vooraleer je naar het volgende deel doorgaat. Niet overslaan dus. Want html, css en JavaScript is oefenen, oefenen en oefenen. Alleen zo word je de heerser van html-illustratie!



Illustratie 1 – Check aan het einde van dit deel of je een antwoord kan geven op deze doelstellingen. [\[soopbooks.com\]](http://soopbooks.com)

1. Je kent de voor- en nadelen van de canvas.
2. Je weet dat de canvas een rasterafbeelding genereert.
3. Je kan een geldige html5-structuur opbouwen.
4. Je kan een canvas-container in de body aanmaken.
5. Je kan een id aan een html-element toekennen.
6. Je kan het begrip context omschrijven.
7. Je kan een context aanmaken.
8. Je weet hoe een object in elkaar zit.
9. Je kan de context bewaren en ophalen.
10. Je kan het concept van een path toelichten.
11. Je kan een path creëren en afsluiten.
12. Je kan teksten op de context neerschrijven.
13. Je kan de kleur en het lettertype van tekst op de context aanpassen.
14. Je kan tekst op de context uitlijnen.
15. Je kan de basislijn van tekst op de context aanpassen.
16. Je kan vulkleuren, verlopen en patronen gebruiken.
17. Je kan omlijningen toepassen.
18. Je kan een lijn en een kromme op de context tekenen.
19. Je kan lijnen samenvoegen.

20. Je kent het verschil tussen een boog, een bázier-kromme en een kwadratische boog.
21. Je kan figuren zoals een rechthoek en een cirkel tekenen.
22. Je kan een deel van een rechthoek uitsnijden.
23. Je kan een deel van een cirkel op de context plaatsen.
24. Je kan met clipping maskers aanmaken.
25. Je kan figuren schalen, roteren of vervormen.
26. Je kan externe afbeeldingen laden en in de context tonen.
27. Je kan de grootte van externe afbeeldingen wijzigen en uitsnijdingen maken.
28. Je kan schaduwen aan een figuur toevoegen.
29. Je kan figuren transparant maken.
30. Je kan figuren als een extern bestand opslaan.



Career @ total.nl

In Nederland bestaat de Total Groep uit meer dan **10 bedrijven** die bij elkaar ruim **2.500 medewerkers** in dienst hebben. Zij houden zich bezig met **exploratie en productie van aardgas, raffinage van ruwe olie**, de verkoop van **olieproducten en chemie**. Total produceert op een **verantwoorde manier energie en producten** die nodig zijn in onze maatschappij, daarbijlettend op de **besparing van energie en de bescherming** van het milieu. Vanuit het kantoor in **Den Haag** verkopen wij brandstoffen en smeermiddelen. Dit in een **uitdagende markt** waarin wij als organisatie continu moeten **ontwikkelen en vernieuwen**.

Deze markt vraagt om **getalenteerde medewerkers!**

Check www.total.nl



Total kies je niet toevallig **TOTAL**



Click on the ad to read more

2 Inleiding

Met het `<canvas>`-html-element bouw je met JavaScript-code fijne illustraties en animaties. Dat lijkt op het eerste zicht misschien wat vreemd. Illustraties en animaties bouw je toch met een grafisch programma en bewaar je daarna als een .jpg, .png of .gif? Toch kan het ook met html en JavaScript. Maar waarom?

2.1 Grootte bestanden

Waarom zou je een illustratie met code opbouwen als dat makkelijk met een tekenpakket kan? De omvang van grafische bestanden is al een mooie reden. Ook al werken surfers vandaag veelal met stevige breedbandverbindingen, toch surfen mobiele gebruikers met hun smartphone niet altijd tegen even hoge snelheden. Een illustratie die uit zuivere tekstopdrachten bestaat, flits sneller over de internetsnelweg, dan een volwaardig afbeeldingbestand.

Toch staat niet alleen de overdrachtsnelheid voorop. Er is meer. Zo verwerkt een browser JavaScript-code sneller dan wanneer diezelfde browser een afbeelding moet inladen. Bij een externe afbeelding moet de browser het bestand namelijk eerst aanspreken om daarna de inhoud ervan te bepalen. Zijn de afbeeldinggegevens bijvoorbeeld gecodeerd volgens het jpg-algoritme of volgens de png-afspraken? Vervolgens moet de browser de afbeelding uitpakken – vele afbeeldingformaten comprimeren de informatie – om die gegevens tot slot naar het scherm te schrijven. JavaScript-code hoeft alleen maar gelezen en uitgevoerd te worden.

2.2 Bewerking en interactiviteit

Maar er is nog meer. Binnen de `<canvas>` kan je ook afbeeldingen zoals .jpg's of .png's laden én bewerken. Zo zet je met wat code een filter over die afbeelding heen... Tot slot is een `<canvas>` ook boeiend zodra je bijvoorbeeld een spelletje wil ontwikkelen. De tandem canvas en JavaScript zorgt ervoor dat je interactiviteit aan die illustraties kan toevoegen.

2.3 Rasterafbeeldingen

Verder hou je steeds in je achterhoofd dat het `<canvas>`-element voor rasterafbeeldingen zorgt. Daardoor werk je op basis van losse pixels. Zodra je op die canvas inzoomt, krijg je automatisch kwaliteitsverlies. Wil je vectorieel op basis van objecten en wiskundige formules werken, kies dan voor `svg` (zie Deel 3 – `Svg`). Daarmee kan je naar hartenlust schalen.

3 Canvas-container

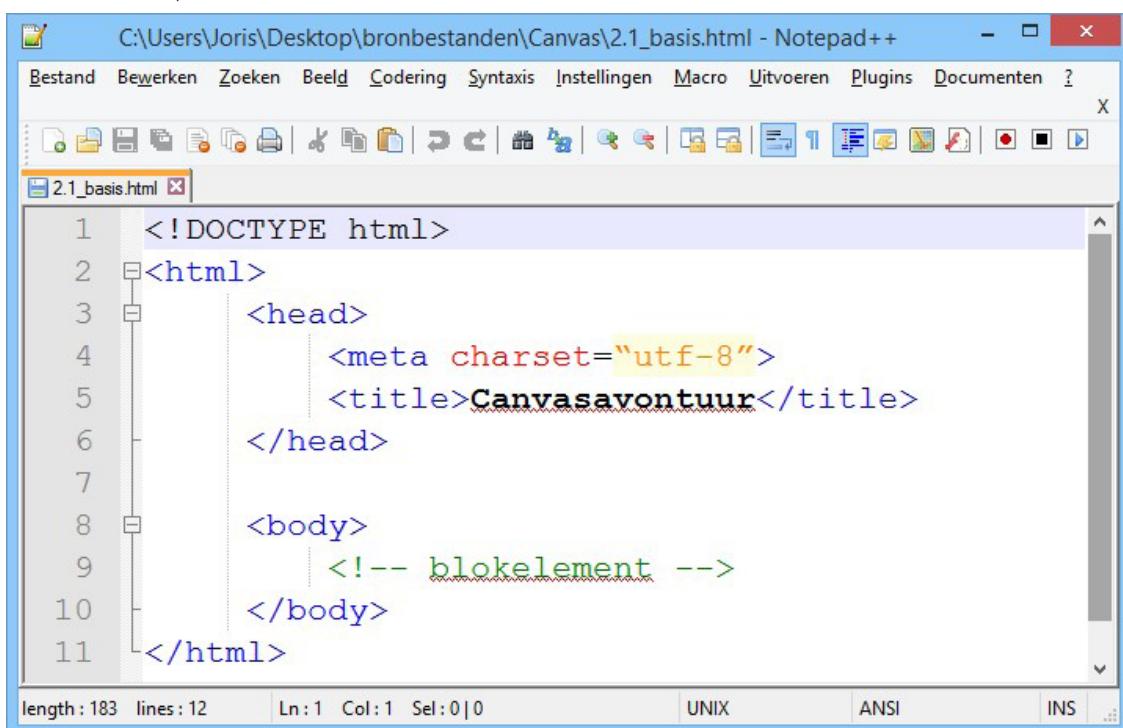
Fijn! Je bent nu helemaal klaar om je eigen illustraties in mekaar te knutselen. Maar hoe begin je nu aan zo'n canvas-avontuur? Eerst en vooral heb je een geldige html5-basisstructuur nodig. Daarna voeg je een <canvas>-container aan de <body> toe en geef je enkele eigenschappen mee. We overlopen die stappen één voor één.

3.1 Html5-structuur

Codeer eerst een basisstructuur in html5.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title></title>
    </head>

    <body>
        <!-- blokelement -->
    </body>
</html>
```



2.1-basis.html in Notepad++

In het bovenstaande voorbeeld zie je insprongen en witregels. Die zijn technisch helemaal niet verplicht, toch zullen insprongen en witruimte je code overzichtelijker maken. Daardoor kan je je code efficiënter lezen én speur je fouten sneller op.

3.2 Canvas in body

Vervolgens voeg je de <canvas>-container in de <body> in. Dat doe je binnen de html-flow, of met andere woorden, daar op je pagina waar je de canvas op de webpagina wil tonen.

```
<body>
    <canvas></canvas>
</body>
```

Die canvas vormt eigenlijk de plaats op een webpagina waar je illustraties of animaties wil tonen. Vergelijk het met een schildersdoek dat tegen de muur hangt. De kunstenaar werkt alleen op het doek en laat de muur met rust.

HTML5 en CSS3

Zit html5 je niet meer zo fris in de vingers? Grasduin dan misschien eerst even in het boek HTML5 en CSS3 van dezelfde auteur.

**STUDY ENGINEERING AND SCIENCE
AT AALBORG UNIVERSITY IN DENMARK**

AIM FOR A BRIGHT FUTURE - MOVE THE WORLD!

- Highly ranked programmes
- Problem based learning and team work
- Close collaboration with industrial partners

Study Engineering and Science in Denmark – build the future and move the world!

APPLY NOW - DEADLINE 1 APRIL 2015

MORE INFORMATION ON MOVETHEWORLD.DK

**AALBORG UNIVERSITY
DENMARK**

**NO TUITION
FEE FOR
EU/EEA-CITIZENS**

Download free eBooks at bookboon.com

Een `<canvas>`-element vormt een zuivere html-container. Wanneer je geen specifieke breedte en hoogte voor je canvas opgeeft, maakt de browser daar standaard 300 pixels breed op 150 pixels hoog van. Echter kan je zelf ook eigen waarden met de eigenschappen `width` en `height` toekennen.

```
<body>
    <canvas width="600px" height="600px"></canvas>
</body>
```

In het bovenstaande voorbeeld kiezen we voor een vierkant formaat. Niets houdt je echter tegen om een andere vorm te ontwerpen. De vorm van de canvas is namelijk sterk afhankelijk van de inhoud die je wil tonen.

Dom-object

De `<canvas>` vormt een html-dom-object (document object model). Dat betekent dat die `<canvas>`-container niet alleen de zichtbare gegevens – zoals de naam `canvas` en de eigenschappen `width` en `height` – bevat, ook vele andere eigenschappen en methodes hangen eraan vast. Denk maar aan de globale eigenschappen zoals `id`, `draggable` of `title` en aan methodes zoals `onclick`, `onload` en `onerror`.

3.3 Identiteit

We zijn bijna klaar met het html-document. Voeg nu nog één belangrijke eigenschap in, namelijk `id`. Die eigenschap zullen we gebruiken om het canvas-dom-object te koppelen aan een JavaScript-object.

```
<body>
    <canvas id="canvas" width="600px"
            height="600px"></canvas>
</body>
```

Aan de html-zijde ben je nu rond. Alle volgende codes schrijf je verder in JavaScript. Dat kan je binnen het bovenstaande html-document doen, toch valt dat af te raden. Kies er liever voor om de html van de JavaScript te scheiden. Zo zal je niet alleen efficiënter werken, ook vermijd je fouten.

Daarom werken we in dit boek in lagen. Naast een html-bestand – dat uitsluitend html bevat – maken we een apart JavaScript-bestand aan. Dat bestand, `.js`, bevat uitsluitend JavaScript-code. Laten we het `.js`-bestand eerst in de `<head>` van het html-bestand nog koppelen.

```
<script type="text/javascript" src=".//js/tekenen.js"></script>
```

Wanneer je gewoon bent om externe `.css`-bestanden te koppelen, had je misschien spontaan aan de `<link>`-tag gedacht. Een JavaScript-bestand koppel je echter met de bekende `<script>`-container en het source `src`-argument. Voeg eventueel een path naar een submap toe. Dat laatste is niet vereist, toch getuigt het van een goede programmeerpraktijk.

Download free eBooks at bookboon.com

3.4 Opmaak

Wens je dat de <canvas> opgemaakt wordt op je webpagina? Dat is een taak voor css, waarbij die bovenstaande id dan ook van pas komt. id is namelijk ook een css-selector.

Maak een extern .css-bestand aan. Toegegeven, je kan dat ook probleemloos binnen het .html-document zelf, toch blijft het verstandig om in lagen te programmeren. We maken daarvoor canvas.css aan.

```
/* canvas.css */
canvas {
    border: 1px solid blue;
}
```



3.4-canvas.css in Notepad++

In de bovenstaande code zorgen we ervoor dat het tekenvlak, de canvas, met een blauwe lijn omrand wordt. Zo zie je in één klap de afmetingen van je grafische werkvlak.

Lagen en programmeren

Zit css3 je niet meer zo fris in de vingers? Grasduin dan misschien eerst even in mijn boek HTML5 en CSS3. Is het lagenconcept je niet meer duidelijk? Loop dan even door mijn boek JavaScript en JQuery.

3.5 Samenvatting

Even samenvatten. De onderstaande html-code is een prima sjabloon om bij al je .js-bestanden te gebruiken. Het volstaat om de bestandsnaam van je .js-bestand aan te passen en klaar.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Canvasavontuur</title>
        <script type="text/javascript" src="bestand.js">
        </script>
        <link href="canvas.css" rel="stylesheet"
              type="text/css">
    </head>

    <body>
        <canvas id="canvas" width="600px" height="600px">
        </canvas>
    </body>
</html>
```

The screenshot shows the Notepad++ application window with the file '2.1_canvas.html' open. The code is displayed in a syntax-highlighted format. The 'head' section contains meta charset, title, script, and link tags. The 'body' section contains a canvas element with width and height attributes. The Notepad++ interface includes a toolbar, menu bar, status bar at the bottom, and a vertical scroll bar on the right.

2.1-canvas.html in Notepad++

4 Context

4.1 Gereedschapskist

Een klassieke schilder die zich op een schildersdoek wil uitleven, zorgt eerst voor penselen, kwasten, verf en ander creatief materiaal. Ook een digitale kunstenaar moet eerst het nodige digitale gereedschap voorzien. Dat zit in de zogeheten context van de <canvas> vervat. Je kan die context vergelijken met een gereedschapskist vol digitale penselen met verschillende diktes, diverse kleuren verf, plakletters, meetlatten, ...

4.1.1 Stappen

De context roep je in JavaScript als volgt op.

```
var c, ctx;  
c = document.getElementById("canvas");  
ctx = c.getContext("2d");
```

A photograph of a child wearing a white space suit and helmet, standing next to a large, metallic, cylindrical object resembling a rocket or a mobile observatory. The child is positioned in a grassy field with trees in the background. A large, stylized orange arrow points diagonally across the image, from the word "future" on the left to "present" on the right. The "accenture" logo is visible in the bottom right corner of the advertisement.

future > present

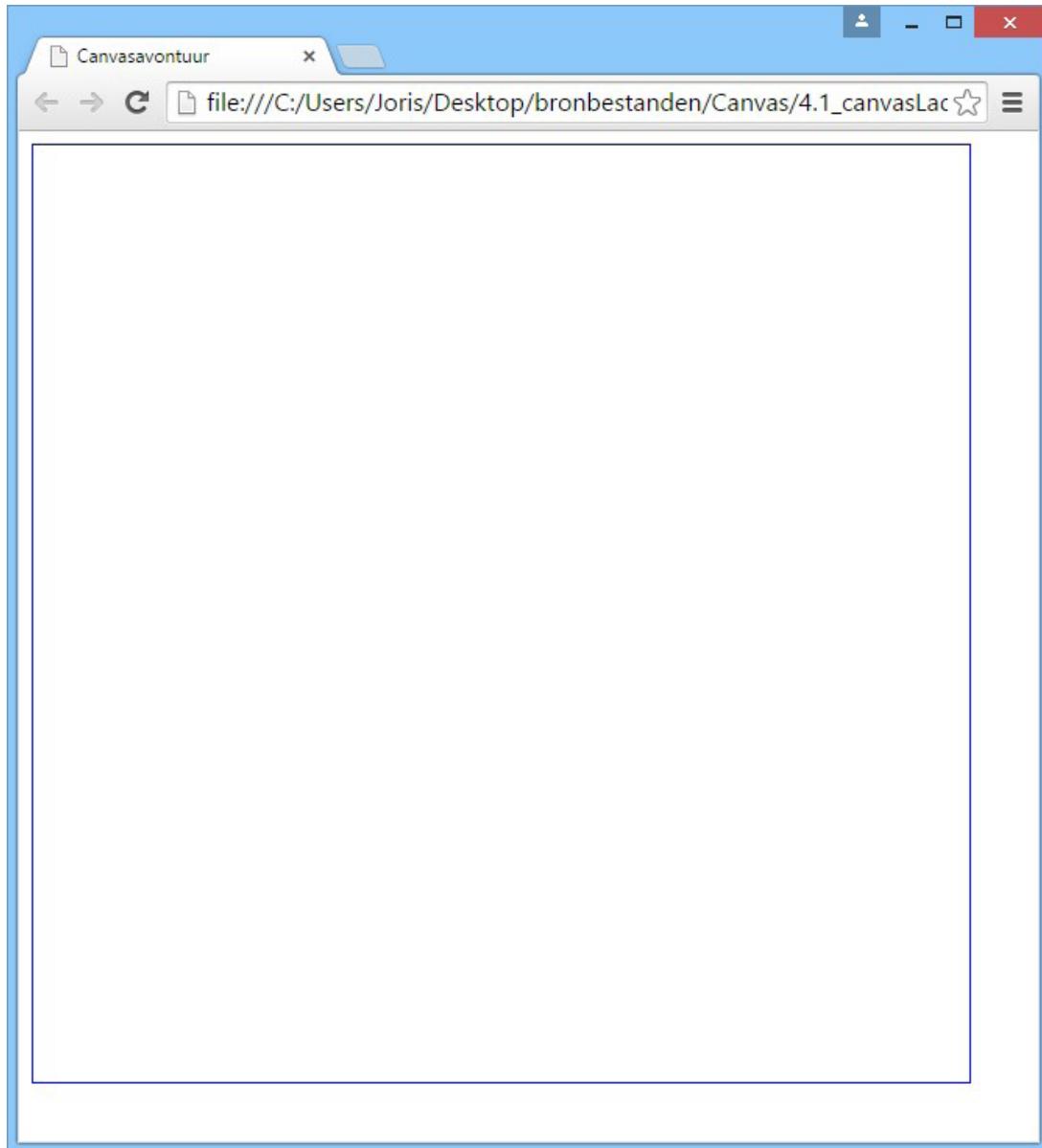
Be greater than.

Dit is jouw kans om bij een organisatie te werken die je meer mogelijkheden, meer uitdaging en meer voldoening biedt.

Ga naar accenture.nl/carriere

2014 Accenture. All rights reserved.





4.1.1-canvasLaden.js in Google Chrome

Laten we eerst de bovenstaande code even ontleden. Begin met de declaratie van twee enkelvoudige variabelen.

```
var c, ctx;
```

De variabele `c` staat voor *canvas*, terwijl `ctx` de afkorting van *context* inhoudt. Uiteraard mag je die variabelen ook anders noemen, toch is het onder programmeurs een ongeschreven afspraak om die variabelennamen `c` en `ctx` te gebruiken. Vergeet namelijk nooit dat het internet je grote vriend is, die dag en nacht paraat staat. Andere programmeurs posten er namelijk talloze voorbeelden op, waarbij ze die variabelennamen gebruiken.

Daarna initialiseer je de eerste variabele.

```
c = document.getElementById("canvas");
```

Na het gelijkheidsteken zie je de waarde staan die je in de variabele c stopt.

```
document.getElementById("canvas")
```

Met de JavaScript-opdracht `getElementById` ga je lijn per lijn de volledige html-code van het bestand `2.1_canvas.html` – het zogeheten `document` – doorzoeken. Die zoekfunctie speurt naar het html-element dat het opgegeven `id` bezit. Let daarbij op de diverse hoofdletters in de naam van die zoekfunctie, spel je de naam van de functie verkeerd, dan werkt je JavaScript-code voor geen meter.

```
getElementById("canvas")
```

In onze code zoeken we naar een html-element met het `id` `canvas`. De functie zal dan het volledige `<canvas>`-element uit `2.1_canvas.html` vinden, zoals je hieronder ziet.

```
<canvas id="canvas" width="600px" height="600px"></canvas>
```

In the past four years we have drilled
89,000 km
That's more than **twice** around the world.

Who are we?
We are the world's largest oilfield services company¹. Working globally—often in remote and challenging locations—we invent, design, engineer, and apply technology to help our customers find and produce oil and gas safely.

Who are we looking for?
Every year, we need thousands of graduates to begin dynamic careers in the following domains:

- Engineering, Research and Operations
- Geoscience and Petrotechnical
- Commercial and Business

What will you be?

Schlumberger

¹Based on Fortune 500 ranking 2011. Copyright © 2015 Schlumberger. All rights reserved.

Toch vindt `getElementById()` meer dan dat. Het gevonden element `<canvas>` vormt binnen html een object, met eigenschappen en methodes – we drukken er nog een keertje op, objecten zijn nu eenmaal een belangrijke bouwsteen bij programmeren. Ook de niet-zichtbare eigenschappen en methodes van dat element – vaak met de standaardwaarde – worden daarbij gevonden.

Het zoekresultaat bevat dus heel wat gegevens. Daarom converteert JavaScript de enkelvoudige variabele `c` automatisch naar een object. En dat heeft op zijn beurt weer gevolgen. Daardoor kan je in de onderstaande lijn code namelijk weer een methode aan die variabele koppelen met de member access operator punt.

```
ctx = c.getContext("2d");
```

Je initialiseert de variabele `ctx` met de methode `getContext()` van de canvas. Als argument geef je daarbij de optie `2d` mee. Zo kan je vlakke figuren ontwerpen. Wil je in `3d` werken, dan vervang `2d` door `webgl`. Echter behandelt dit ebook alleen `2d`, voor `3d` is er namelijk nog geen volledige ondersteuning bij de browsers.

4.1.2 Extern

Wanneer je met een extern `.js`-bestand werkt, mag je niet vergeten om de bovenstaande code pas uit te voeren nadat de volledige html-code uit het html-document door de browser geladen is.

```
//code voor een extern JavaScript-bestand
window.onload = tekenen; //pas uitvoeren nadat de <body> geladen is

function tekenen() {

    //verwijzing naar een html-element met id canvas
    var c, ctx;
    c = document.getElementById("canvas");
    ctx = c.getContext("2d");
}
```

Wanneer je een extern JavaScript-bestand vanuit de `<head>` in je html-bestand met de `<script>`-container laadt, wordt de inhoud van dat externe bestand meteen uitgevoerd. In die code laat je de `getElementById`-methode naar een `id` met als waarde `canvas` zoeken. Echter is op dat moment dat element nog niet geladen. Je zit in de flow namelijk nog maar aan de `<head>`.

Daarom stop je al je tekenwerk in één of meerdere aparte functies, zoals de functie `tekenen()`. Die functie laat je niet meteen bij het laden van de JavaScript-code uitvoeren. Je wacht met de aanroeping ervan totdat de `<body>` van de html volledig geladen is. Daarvoor zorgt de `window.onload`.

4.2 Bewaren en ophalen

Verderop zal je leren hoe je alle gereedschappen naar je hand kan zetten. Zo kan je een kleur instellen, het soort lijn dat je trekt, ... Al die instellingen behoren tot de gereedschapskist, de context. Die waarden kan je tijdelijk bewaren. Op die manier kan je al je instellingen opslaan, vervolgens met een gerust hart andere kleuren selecteren, een andere lijndikte, een nieuw lettertype, enzovoort even wijzigen, om daarna de voorgaande instellingen in één keer terug op te halen.

```
ctx.fillStyle = "#F00"; //rode vulkleur
ctx.fillRect(0,0,50,50); //rode rechthoek tekenen
ctx.save(); //huidige vulkleur en andere waarden bewaren
ctx.fillStyle = "#0F0"; // groene vulkleur
ctx.fillRect(100,150,75,110); //groene rechthoek tekenen
```

In de bovenstaande code gebruik je eerst een rode vulkleur, die je vervolgens bewaart met de opdracht `save()`. Vervolgens wijzig je de kleur naar groen.

The advertisement features a young man with curly hair smiling, wearing a backpack and holding a coffee cup. The background is a blurred indoor setting. The text is overlaid on the right side of the image.

**AARHUS
BSS** SCHOOL OF BUSINESS AND SOCIAL SCIENCES
AARHUS UNIVERSITY

AACSB ACCREDITED **AMBA ACCREDITED** **EFMD EQUIS ACCREDITED**

**Bachelor of Engineering -
Global Management
and Manufacturing**

Aarhus BSS is part of Aarhus University in Denmark. It is ranked among the top 100 universities in the world due to its high standards in both education and research.

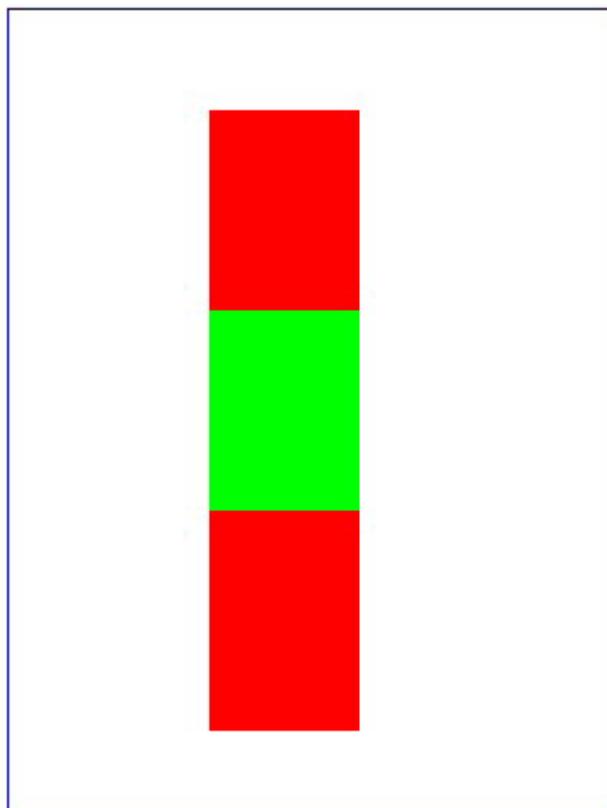
We offer English-taught programmes at all educational levels: Bachelor's, Master's, continuing education (MBA) and PhD programmes.

Read more
bss.au.dk/international



Wil je nu snel terugkeren naar de rode verf? Dan kan je de `restore()`-methode op de context toepassen.

```
ctx.fillStyle = "#F00"; //rode vulkleur  
ctx.fillRect(0,0,50,50); //rode rechthoek tekenen  
ctx.save(); //huidige vulkleur en andere waarden bewaren  
ctx.fillStyle = "#0F0"; // groene vulkleur  
ctx.fillRect(100,150,75,110); //groene rechthoek tekenen  
ctx.restore(); //keer terug naar de vorige instellingen  
ctx.fillRect(100,250,75,110); //terug een rode rechthoek
```



4.2_canvasBewarenEnOphalen.js

Met de `restore()`-methode haal je in één klap alle eigenschappen van de vorige context terug op, zoals die op het moment van de `save()`-methode ingesteld stonden.

5 Paden

Paden vormen bij het tekenen van illustraties een belangrijk begrip. Werk je regelmatig in een vectoriëel tekenpakket, dan klinkt die term je alvast niet onbekend in de oren. Paden zijn namelijk het samen nemen van diverse vormen, zoals drie lijnen die met één pad een driehoek vormen.

Is het echter niet vreemd dat een rasteromgeving zoals de canvas met paden aan de slag gaat? Misschien wel, toch vormen paden ook bij deze bitmaps een krachtig codehulpmiddel om vormen weer te geven.

5.1 Begrip

Zo'n pad zorgt ervoor dat een rechte lijn een samenhangend geheel van allemaal punten tussen een beginpunt en een eindpunt vormt. Dat is voor een processor sneller om te tekenen, dan wanneer de computer de coördinaten van elk individueel punt uit die lijn zou moeten opslaan. Met een pad volstaat het om de coördinaten van het beginpunt en van het eindpunt te onthouden, alle andere punten kunnen daarvan afgeleid – lees berekend – worden.

De canvas maakt een onderscheid tussen een path en een subpath. Een path is een aaneenschakeling van figuren, zoals lijnen en krommen. Een individuele figuur – bijvoorbeeld één enkele lijn – is een subpath. Een path kan ook een aaneenschakeling van subpaden zijn.

5.2 beginPath()

Voor we met een lijn aan de slag gaan, moet je eerst een pad maken. Zo'n pad – een *path* in het Engels – zorgt ervoor dat je samen horende tekeninstructies kan koppelen.

```
ctx.beginPath();
```

Het volstaat om de context-methode `beginPath()` zonder argumenten aan te roepen. JavaScript regelt achter de schermen alle stappen voor jou om een pad aan te maken.

Laten we eens een stukje code bekijken. Probeer hiervan alleen de grote lijnen te volgen, de details over hoe je bijvoorbeeld een lijn tekent, hoe je een vulkleur kan bepalen en dergelijke, kom je later nog tegen.

```
ctx.beginPath(); // begin een subpath
ctx.strokeStyle = "#A00"; //vulkleur rood
ctx.lineWidth = 12; //lijndikte 12 pixels
ctx.moveTo(20,10); //begin de lijn op coördinaat (20,10)
ctx.lineTo(20,100); //trek de lijn naar coördinaat (20,100)
ctx.stroke(); //teken de lijn
```

```
ctx.beginPath(); //begin een nieuw subpath
ctx.strokeStyle = "#0A0"; //vulkleur grijs
ctx.lineWidth = 24; //lijndikte 24 pixels
ctx.moveTo(50,10); // begin de lijn op coördinaat (50,10) ctx.
lineTo(50,100); //trek de lijn naar coördinaat (50,100) ctx.
stroke(); //teken de tweede lijn
```

**5.2_beginPath.js**

De realisatie van de A2 landtunnel. Het volledige energiesysteem van het Oosterdokseiland. De projecten van Cofely zijn al snel bijzonder. Als grootste technisch dienstverlener van Nederland kun je bij ons innovatieve oplossingen creëren voor bijvoorbeeld Shell, ASML, Heineken en Rijkswaterstaat.

Of je nu technisch specialist bent en je vakinhoudelijk wilt verdiepen, of jezelf in een leidinggevende rol wilt ontwikkelen: binnen onze veelzijdige organisatie kies je zelf welke kant je op wilt.

Cofely is groot maar persoonlijk, internationaal en toch dichtbij. Heb jij een passie voor techniek en de drive om het beste uit jezelf te halen? Ontdek dan hoe ver jij kunt komen op werkenbijcofely.nl.

Verbind jezelf aan **COFELY**
GDF SUEZ



De eerste lijn uit het bovenstaande voorbeeld geeft de start van een nieuw pad weer.

```
ctx.beginPath(); // begin een subpath
```

Dat betekent dat alle gereedschappen (terug) op hun standaardwaarden vallen. Zo zal de lijndikte bijvoorbeeld op waarde 1 pixel ingesteld worden of bevat de verfpot de kleur zwart.

Daarna zie je hoe voor het nieuwe pad echter met #A00 de kleur rood gekozen wordt en voor een lijndikte van drie pixels geopteerd wordt. Alle figuren die je dan tekent, krijgen die dikte mee.

```
ctx.strokeStyle = "#A00"; //vulkleur rood  
ctx.lineWidth = 3; //lijndikte 3 pixels
```

Vervolgens zet je de tekenpen klaar op de coördinaat (10, 10) en bepaal je het eindpunt van de lijn, namelijk (10, 100).

```
ctx.moveTo(10,10); //begin de lijn op coördinaat (10,10)  
ctx.lineTo(10,100); //trek de lijn naar coördinaat (10,100)
```

Tot slot maak je de lijn zichtbaar op de canvas met de methode `stroke()`.

```
ctx.stroke(); //teken de lijn
```

Vervolgens lees je hoe er een nieuw subpad gecreëerd wordt met `beginPath()`. Opnieuw voert de context een reset van alle tekeneigenschappen uit, zoals de kleurwaarden, de lijndiktes, ...

```
ctx.beginPath(); //begin een nieuw subpath
```

Daarna zie je hoe de kleur van het tweede pad naar donker grijs omgezet wordt. De lijn wordt met 6 pixels bovendien ook dikker.

```
ctx.strokeStyle = "#555"; //vulkleur grijs  
ctx.lineWidth = 6; //lijndikte 6 pixels
```

Tot slot geef je aan de context de opdracht om die tweede lijn ook effectief te tekenen.

```
ctx.stroke(); //teken de tweede lijn
```

Kortom, een `beginPath()` zorgt ervoor dat al je tekengerief terug naar de fabrieksinstellingen keert én dat alle punten netjes samen horen.

5.3 closePath()

Met `closePath()` sluit je alleen een bestaand subpath af. Een path hoeft je niet dicht te gooien. De bovenstaande code zou je dus als volgt kunnen aanvullen.

```
ctx.beginPath(); // begin een subpath  
ctx.strokeStyle = "#A00"; // vulkleur rood  
ctx.lineWidth = 12; // lijndikte 12 pixels  
ctx.moveTo(20,10); // begin de lijn op coördinaat (20,10)  
ctx.lineTo(20,100); // trek de lijn naar coördinaat (20,100)  
ctx.stroke(); // teken de lijn  
ctx.closePath(); // subpath afsluiten  
  
ctx.beginPath(); // begin een nieuw subpath  
ctx.strokeStyle = "#0A0"; // vulkleur grijs  
ctx.lineWidth = 24; // lijndikte 24 pixels  
ctx.moveTo(50,10); // begin de lijn op coördinaat (50,10)  
ctx.lineTo(50,100); // trek de lijn naar coördinaat (50,100)  
ctx.stroke(); // teken de tweede lijn  
ctx.closePath(); // subpath afsluiten
```



5.3_closePath.js

Pas je `closePath()` toe, dan zal het volgende `beginPath()` starten bij het laatste punt van het vorige pad.

Je merkt in het bovenstaande voorbeeld dat er geen verschil bestaat in uitvoer tussen 5.2_beginPath.js en 5.3_closePath.js. Moet je die methode dan toepassen? Klaarblijkelijk sluit JavaScript een vorig subpath sowieso af, zodra je een nieuw subpath aanmaakt. Echter getuigt het gebruik van `ctx.closePath();` van een goede scriptpraktijk. Leer jezelf zulke gewoontes aan, zo leer je correct en efficiënt programmeren.

6 Teksten

Ook al teken je in de canvas bergen illustraties, zo nu en dan zal je ook met tekst aan de slag moeten gaan. Echter is de canvas geen tekstverwerker. Integendeel, je zal teksten op de canvas tekenen in plaats van schrijven. Gelukkig heeft de context heel wat gereedschappen aan boord om je daarbij te helpen.

Daarom biedt de context je methodes en eigenschappen aan voor lettertypes, lettergroottes, kleuren, uitlijnen en de basislijn. We overlopen ze één voor één.

6.1 Lettertype en grootte

Syntax

Een lettertype of font stop je in de context-eigenschap `font`. Vergis je echter niet met de css-eigenschap `font-family`. De context-eigenschap kan namelijk vél meer informatie bevatten dan zijn css-tegenhanger. Zo voeg je ook stijlen en grootte toe.

```
ctx.font = "stijl grootte fontnaam";
```

ANTEA GROUP

Van stad tot land, van water tot lucht; de ingenieurs en adviseurs van Antea Group dragen sinds jaar en dag bij aan onze leefomgeving. We ontwerpen bruggen en wegen, realiseren woonwijken en waterwerken. Maar we zijn ook betrokken bij thema's zoals milieu, veiligheid, assetmanagement en energie.

Wie zoeken wij? Het werk van Antea Group is enorm veelzijdig. We tekenen en rekenen, onderzoeken en adviseren, organiseren en realiseren. Dit doen we binnen verschillende werkvelden. Of je nu starter bent of senior, specialist of allrounder: in onze organisatie zijn er altijd mogelijkheden. We zoeken naar hbo'ers en wo'ers met oog voor de wereld om zich heen en passie voor hun werk.

WWW.WERKENBIJANTEAGROUP.NL

Knappe mensen. Mooi bedrijf.

anteagroup



Click on the ad to read more

Je ziet hoe je drie eigenschappen in één enkele waarde verenigd. Eerst geef je de stijl weer, echter mag je die ook gerust weg laten. Dan staat je lettertype in de regular-stijl.

```
ctx.font = "grootte fontnaam"; // zonder stijlaanduiding
```

Vervolgens druk je de grootte van de letters in pixels uit. Ook die kan je weglaten, dan gebruikt de context simpelweg de standaardgrootte van 10 pixels.

```
ctx.font = "fontnaam"; // zonder stijlaanduiding en grootte
```

Kortom, zo je wil je de `font`-eigenschap beperken tot de naam van een lettertype. Wanneer je zelfs die zou weglaten, dan kiest de context standaard voor een sans-serif-lettertype (schreefloos).

Wat omvat `font` aan mogelijkheden? Je begint eerst met de stijl van het lettertype, zoals `normal`, `oblique`, `italic` of `bold`. Ben je vanuit de typografie met termen zoals `regular` vertrouwd? Helaas ondersteunt de canvas dat begrip niet, kies `normal` als alternatief.

```
ctx.font = "italic 12px Helvetica";
```

Daarna voeg je de lettergrootte in. Daarbij kan je tussen pixels of punten kiezen. Toch genieten in de digitale wereld de pixels de voorkeur.

```
ctx.font = "italic 12pt sans-serif";
```

Als laatste plaats je de naam van het font.

```
ctx.font = "italic 12pt Impact";
```

Verder kan je ook een lijst van lettertypes toevoegen, net zoals je dat in CSS zou doen. Scheid elk lettertype dan met een komma.

```
ctx.font = "italic 12px Helvetica, Arial";
```

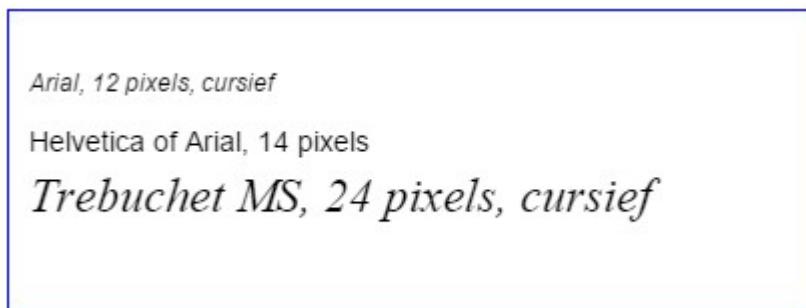
Bestaat de naam van het lettertype uit meerdere woorden – en staat er dus een spatie in de naam – dan moet je aanhalingstekens rond die naamdelen plaatsen. Let er dan wel op dat je dubbele en enkele aanhalingstekens combineert.

```
ctx.font = "italic 12px 'Trebuchet MS' ";
```

Merk je bovendien op dat je de drie onderdelen van de `font`-eigenschap (stijl, grootte en fontnamen) *niet* met een komma scheidt? Je gebruikt een spatie om die van elkaar te onderscheiden.

Voorbeeld

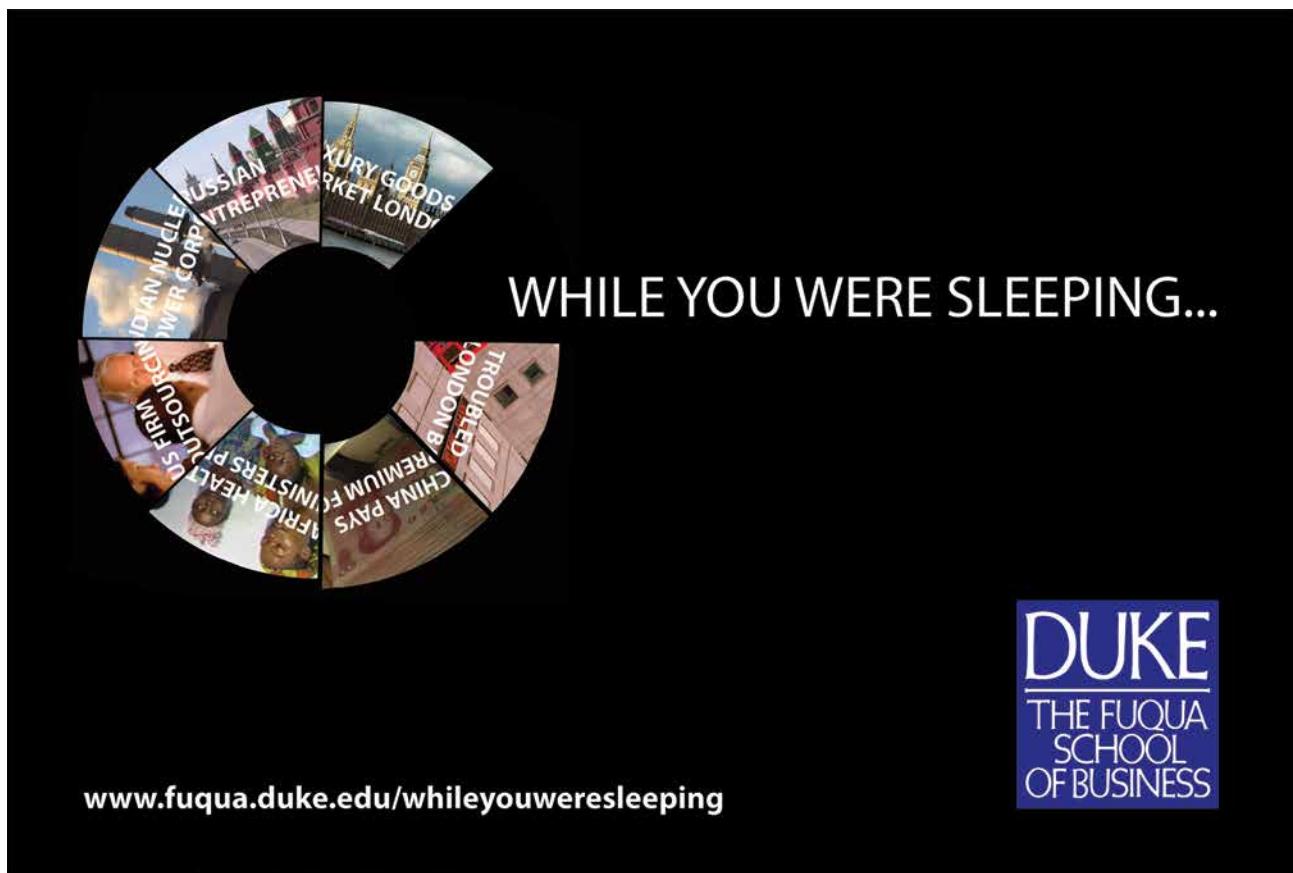
```
ctx.font = "italic 12px Helvetica";
```



6.1_lettertype.js

6.2 Schrijven

Nu de context-eigenschap `font` gevuld is, kan je starten met het schrijven van tekst. Daarvoor beschik je over twee mogelijkheden, `fillText()` en `strokeText()`.



Click on the ad to read more

6.2.1 fillText()

Syntax

Met `fillText()` schrijf je tekst vanaf een startpunt op de canvas neer.

```
ctx.fillText("tekst", x, y, maxBreedte);
```

`tekst` Geef de tekst – tussen aanhalingstekens – die je wil tonen.

`x` De `x`-coördinaat van het startpunt van de basislijn van je tekstvlak.

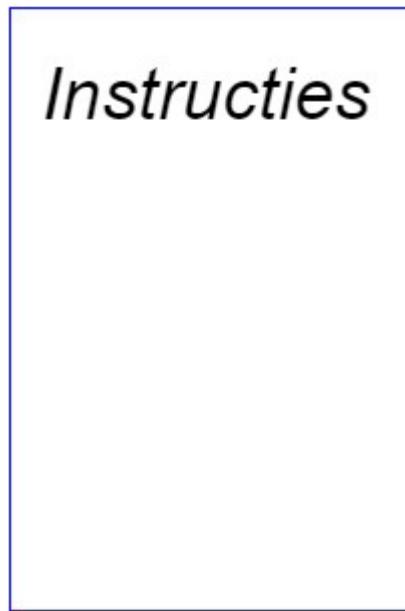
`y` De `y`-coördinaat van het startpunt van de basislijn van je tekstvlak.

`maxBreedte` De maximale breedte uitgedrukt in pixels dat de tekst mag innemen.

Voorbeeld

Met `fillText()` ga je een tekst tekenen die uit opgevulde letters bestaat, zeg maar de klassieke letters.

```
ctx.font = "italic 36px Geneva, sans-serif"; //font-  
//eigenschappen  
ctx.fillText("Instructies", 15, 55); //tekst tekenen
```



6.2.1_fillText.js

Wanneer je tekst schrijft, is het raadzaam om eerst een stijl, een corps en een lettertype te bepalen. Daarna kan je met `fillText()` de tekst op basis van een coördinaat op het scherm positioneren. Je merkt dat het laatste argument, de maximum breedte die de tekst mag innemen, optioneel is.

6.2.2 strokeText()

Syntax

Met `strokeText()` kan je eveneens tekst op het scherm toveren. Echter zijn de letters dan niet ingekleurd, je ziet enkel de omlijning – de zogeheten stroke.

```
ctx.strokeText("tekst", x, y, maxBreedte);
```

`tekst` Geef de tekst – tussen aanhalingstekens – die je wil tonen.

`x` De x-coördinaat van het startpunt van de basislijn van je tekstvlak.

`y` De y-coördinaat van het startpunt van de basislijn van je tekstvlak.

`maxBreedte` De maximale breedte uitgedrukt in pixels dat de tekst mag innemen.

Voorbeeld

```
ctx.font = "italic 36px Geneva, sans-serif";
ctx.strokeText("To illustrate or not to illustrate", 25, 60);
```



To illustrate or not to illustrate

6.2.2_strokeText.js

De methode `strokeText` bezit precies dezelfde argumenten als `fillText`. Ook hier stel je best eerst de stijl, de grootte en het lettertype in, daarna schik je de tekst aan de hand van de coördinaten op de canvas.

6.3 Kleur

Syntax

Een tekst kan je ook een kleur geven. Dan komt de context-eigenschap `strokeStyle` of `fillStyle` om het hoekje loeren.

```
ctx.fillStyle = "kleurwaarde";
```

Download free eBooks at bookboon.com

Voorbeeld

```
ctx.font = "24px Tahoma"; //font instellen
ctx.fillStyle = "orange"; //vulkleur bepalen
ctx.fillText("De digitale wereld",50,50); //tekst schrijven met
//fillText
```



6.3_kleur.js



Winning awards is in our nature

Corbion Purac is proud to be a Top Employer in 2014 for the 5th consecutive year.

Corbion is the global market leader in lactic acid, lactic acid derivatives and lactides, and a leading company in functional blends containing enzymes, emulsifiers, minerals and vitamins. The company delivers high performance biobased products made from renewable resources and applied in global markets such as bakery, meat, pharmaceuticals and medical devices, home and personal care, packaging, automotive, coatings and resins. Its products have a differentiating functionality in all kinds of consumer products worldwide.

Would you like to know more about Corbion Purac or do you want to know more about the vacancies within Corbion Purac? Visit: www.jobsatcorbion.com



Download free eBooks at bookboon.com



Om een kleur aan te duiden, kan je standaard kleurnamen gebruiken, zoals `yellow`, `red`, `green`, `brown`, `aqua`, `sienna`, `deepskyblue`, `maroon`, `tan`, ... Echter is het verstandiger om met een `rgb`-waarde (`red`, `green`, `blue`) te werken. Of beter nog, haal zulke kleurenwaarden uit je externe stylesheet aan de hand van JavaScript op.

```
ctx.fillStyle = "rgb(255,19,31);
```

Zie je dat we de `rgb()`-functie tussen aanhalingstekens plaatsen? Verder kan je ook het transparantiekanaal (alfa) aanspreken.

```
ctx.fillStyle = "rgba(255,19,31, 0.5);
```

Debug-tip

Zorg ervoor dat je zowel een openingsaanhalingsteken typt, als een afsluitend aanhalingsteken. Vergeet je één van beiden, dan werkt je code niet meer.

Tot slot kan je kleuren ook met hexadecimale waarden aanduiden.

```
ctx.fillStyle = "#FF8833";
```

HTML5 en CSS3

Weet je niet meer zo goed hoe `rgb`, `rgba` en die hexadecimale waarden werken? Snuffel dan gerust even in mijn boek `HTML5 en CSS3`.

6.4 Uitlijning

Syntax

Vervolgens wil je de tekst op de canvas kunnen uitlijnen. Daarvoor pas je de context-eigenschap `textAlign` toe. Die kan je zowel bij `fillText()` als bij `strokeText()` gebruiken.

```
ctx.textAlign = "waarde";
```

Deze eigenschap laat de volgende waarden toe: `start`, `end`, `left`, `right` en `center`.

`start` De tekst begint bij het startpunt op de basislijn.

`end` De tekst eindigt vlak voor het startpunt.

`left` De tekst is links uitgelijnd ten opzichte van het startpunt.

`right` De tekst is rechts uitgelijnd ten opzichte van het startpunt.

`center` De tekst is in het midden uitgelijnd ten opzichte van het startpunt.

Voorbeeld

```
ctx.strokeText("Handleiding", 25, 25);  
ctx.textAlign = "start";
```



6.4_uitlijning.js

De tekst Handleiding start nu netjes op coördinaat 25, 25. Merk daarbij op dat we in die code geen `ctx.font`-eigenschap gebruiken. Daardoor val je op de standaardinstellingen voor een lettertype terug, een schreefloos lettertype van 10 pixels groot in een zwarte kleur.

6.5 Basislijn

Met de basislijn bepaal je op welke lijn je tekens geschreven worden. Er bestaan diverse soorten basislijnen, zoals de toplijn, de hangende basislijn, de middenlijn, de alfabetische lijn, de ideografische lijn en de onderlijn. Al die lijnen passen binnen een kader.

Wanneer je verschillende lettertypes naast elkaar plaatst, kan het soms handig zijn om met die basislijn te spelen.

Syntax

De term basislijn is een verzamelnaam. Pas daarom de juiste waarde toe.

```
ctx.textBaseline = "waarde";
```

top Toplijn

hanging Hangende basislijn

middle Middenlijn

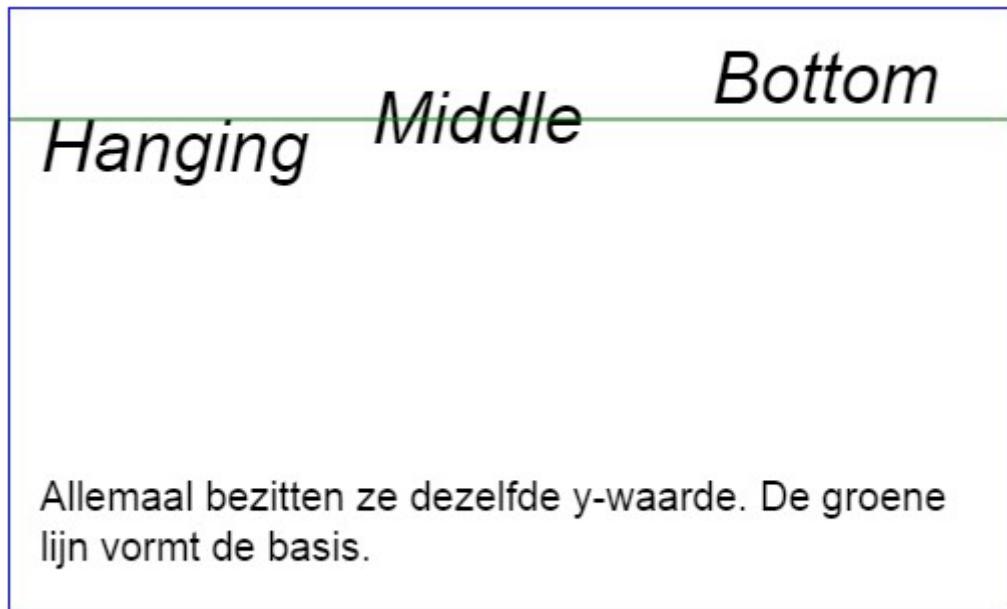
alphabetic Alfabetische lijn

ideographic Ideografische lijn

bottom Onderlijn

Voorbeeld

```
ctx.textBaseline = "hanging";
```



[6.5_basislijn.js](#)

7 Vullen en omlijnen

Ondertussen weet je hoe je tekst kan tekenen. Bij die tekst kan je kiezen uit gevulde letters – `fillText()` – en letters die alleen uit een omlijning bestaan – `strokeText()`. Dat vullen en omlijnen vormt geen zuiver voorrecht van tekst. Ook kan je vormen zoals een rechthoek of een cirkel een vulkleur of een omlijning bieden.

7.1 Vullen

Je kan een object op diverse manieren inkleuren. Dat kan met een egale kleur, een verloop of met een patroon. We zetten die opties op een rij.

7.1.1 Egale vulkleur

Syntax

Met `fillStyle()` bepaal je een enkelvoudige vulkleur.

```
ctx.fillStyle = "kleurwaarde";
```



Julian Lienich, engineer

I can shape the future. Every day.

The E.ON graduate program requires my energy and creative input. In exchange I get to work with up-to-date technologies in a team that supports my professional development. What about you?

Your energy shapes the future.

www.eon-career.com

e-on



Click on the ad to read more

Een kleurwaarde kan een standaard kleurnaam ("red") zijn, een hexadecimale code ("#FF0000") of een rgb(a)-waarde ("rgb(255, 0, 0)").

Met de methode `fill()` pas je de kleur ook effectief toe.

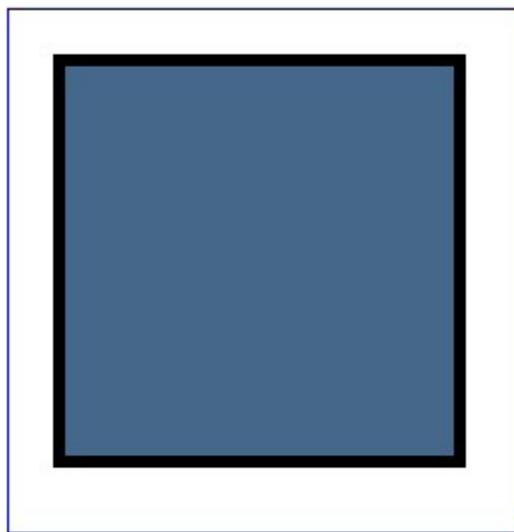
```
ctx.fill();
```

Die methode zorgt namelijk niet alleen voor de vulkleur, ook tekent ze het object zichtbaar op de canvas.

Voorbeeld

Een vorm zoals een vierkant – de details over een rechthoek leer je in een later onderdeel – kan je met één egale kleur vullen.

```
ctx.rect(25,25,200,200); //grootte vierkant bepalen  
ctx.fillStyle = "#456789"; //vulkleur instellen  
ctx.lineWidth = 6; //dikte van de omlijning instellen  
ctx.fill(); //vierkant tekenen met vulkleur  
ctx.stroke(); //kader vierkant tekenen met stroke
```



7.1.1_egaleKleur1.js

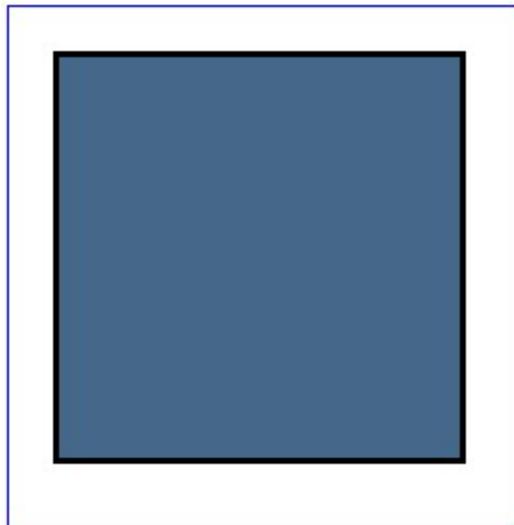
De volgorde lijkt op het eerste zicht misschien wat raar. Je gaat eerst het volledige vierkant met een vulkleur tekenen en pas daarna teken je de omlijning. Echter werk je op die manier met lagen. Laten we het met andere woorden uitleggen. In het bovenstaande voorbeeld teken je eerst een gekleurd vlak ter grote van het vierkant. Vervolgens teken je een lijn op de rand van dat gekleurd vlak.

Het kan echter ook omgekeerd.

```

ctx.rect(25,25,200,200); //grootte vierkant bepalen
ctx.fillStyle = "#456789"; //vulkleur instellen
ctx.lineWidth = 6; //dikte van de omlijning instellen
ctx.stroke(); //kader vierkant tekenen
ctx.fill(); //vierkant vullen met vulkleur

```



7.1.1_egaleKleur2.js

In het bovenstaande voorbeeld teken je eerst de kader. Daarna giet je het gekleurd vlak er over heen. Daardoor zal je kaderlijn onder het vlak terecht komen en zal die niet meer tot weinig zichtbaar zijn.

stroke() én fill()?

Om een figuur zichtbaar op het scherm te krijgen, kan je kiezen. Ofwel gebruik je alleen `stroke()`, ofwel gebruik je uitsluitend `fill()`, ofwel pas je beide toe.

7.1.2 Verloop

Bij een verloop – ofwel *gradient* in het Engels – verandert de startkleur stap voor stap in een eindkleur. Er bestaan diverse soorten verlopen, zoals het lineaire verloop en het radiale verloop.

Syntax lineair

In een lineair verloop gaat een startkleur lijngewijs over naar een eindkleur. Daarvoor bepaal je aan de hand van coördinaten eerst een vlak waarbinnen je het verloop virtueel opbouwt.

```
createLinearGradient(x1, y1, x2, y2);
```

x1 De x-coördinaat van het beginpunt.

y1 De y-coördinaat van het beginpunt.

Download free eBooks at bookboon.com

x2 De x-coördinaat van het eindpunt.

y2 De y-coördinaat van het eindpunt.

Vervolgens moet je nog een beginkleur en een eindkleur definiëren.

```
addColorStop(cijfer, "kleurwaarde");
```

cijfer Een waarde tussen 0 en 1.

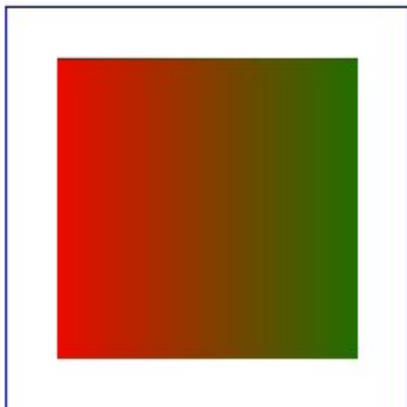
kleurwaarde Een kleurwaarde (kleurnaam, hexadecimale waarde, rgb(a)-waarde).

Voorbeeld lineair

```
var verloop; //variabele declareren
verloop = ctx.createLinearGradient(10,10,200,10); //verloop-
//object aanmaken
verloop.addColorStop(0,"red"); //beginkleur = 0
verloop.addColorStop(1,"green"); //eindkleur = 1
ctx.fillStyle = verloop; //verloop toekennen aan de vulkleur
ctx.fillRect(25,25,150,150); //gevuld vierkant tekenen
```



Download free eBooks at bookboon.com

**7.1.2_verloopLineair.js**

Hoe doe je dat nu? Laten we de bovenstaande code even stukje bij stukje overlopen.

```
var verloop; //variabele declareren
```

Je maakt eerst een variabele aan. Die variabele zal een object inhouden, namelijk het lineaire verloop. Dat definiereer je met de methode `createLinearGradient()`.

```
verloop = ctx.createLinearGradient(10,10,200,10);
```

De coördinaatwaarden (`x1, y1, x2, y2`) vertellen de methode wat het startpunt van de startkleur is en welk punt het eindpunt voor de eindkleur vormt. Die coördinaten staan volledig los van de vorm die je er later mee wenst te vullen. Je definieert er eigenlijk een virtuele ruimte mee, waarbinnen je het verloop opbouwt.

Daarna stel je de begin- en de eindkleur in.

```
verloop.addColorStop(0,"red"); //beginkleur  
verloop.addColorStop(1,"green"); //eindkleur
```

Dat doe je niet binnen de context, wel binnen het eigen `verloop`-object. Daar pas je de methode `addColorStop()` op toe. Vandaar dat je `verloop.` schrijft en niet `ctx.`

Die methode `addColorStop()` vereist twee argumenten. Het eerste argument, de stop, is een getal tussen 0 en 1. Met nul duid je de eerste kleur aan, met 1 bepaal je de laatste kleur. Tussen 0 en 1 kan je wéliswaar kommagetallen gebruiken, maar vergeet niet om binnen JavaScript een punt in de plaats van een komma te schrijven. In totaal kan je theoretisch gesproken dus 101 kleuren in een verloop stoppen.

```
verloop.addColorStop(0,"red"); //eerste kleur
verloop.addColorStop(0.5,"green"); //tweede kleur
verloop.addColorStop(1,"blue"); //derde kleur
```

Het tweede argument van `addColorStop()` bevat de kleurnaam, een hexadecimale waarde of een `rgb`-waarde. Daarin geef je dus de kleur op waarmee het verloop gebouwd moet worden.

Een verloop koppel je tot slot pas in de laatste stap aan één of meerdere vormen.

```
ctx.fillStyle = verloop; //vulkleur instellen met verloop
ctx.fillRect(25,25,150,150); //gevuld vierkant tekenen
```

Hoe doe je dat? Eerst wijs je het verloop aan de eigenschap `fillStyle` toe. Vervolgens gebruik je die eigenschap bij een vorm, zoals een gevulde rechthoek (`fillRect`). Over zulke vormen vind je verderop meer.

Syntax radiaal

Daarnaast kan je ook een cirkelvormig patroon creëren. Dat doe je met de methode `createRadialGradient()`. Zo'n cirkelvormig patroon bouw je aan de hand van twee cirkels op. De binnenste cirkel ken je de startkleur toe, terwijl de buitenste cirkel de eindkleur voor zijn rekening neemt.

Je ontdekt dat de `createRadialGradient()`-methode twee argumenten meer telt dan het lineaire broertje.

```
createRadialGradient(x1,y1,r1,x2,y2,r2);
```

`x1` De x-coördinaat van het middelpunt van de eerste cirkel.

`y1` De y-coördinaat van het middelpunt van de eerste cirkel.

`r1` De straal van de eerste cirkel.

`x2` De x-coördinaat van het middelpunt van de tweede cirkel.

`y2` De y-coördinaat van het middelpunt van de tweede cirkel.

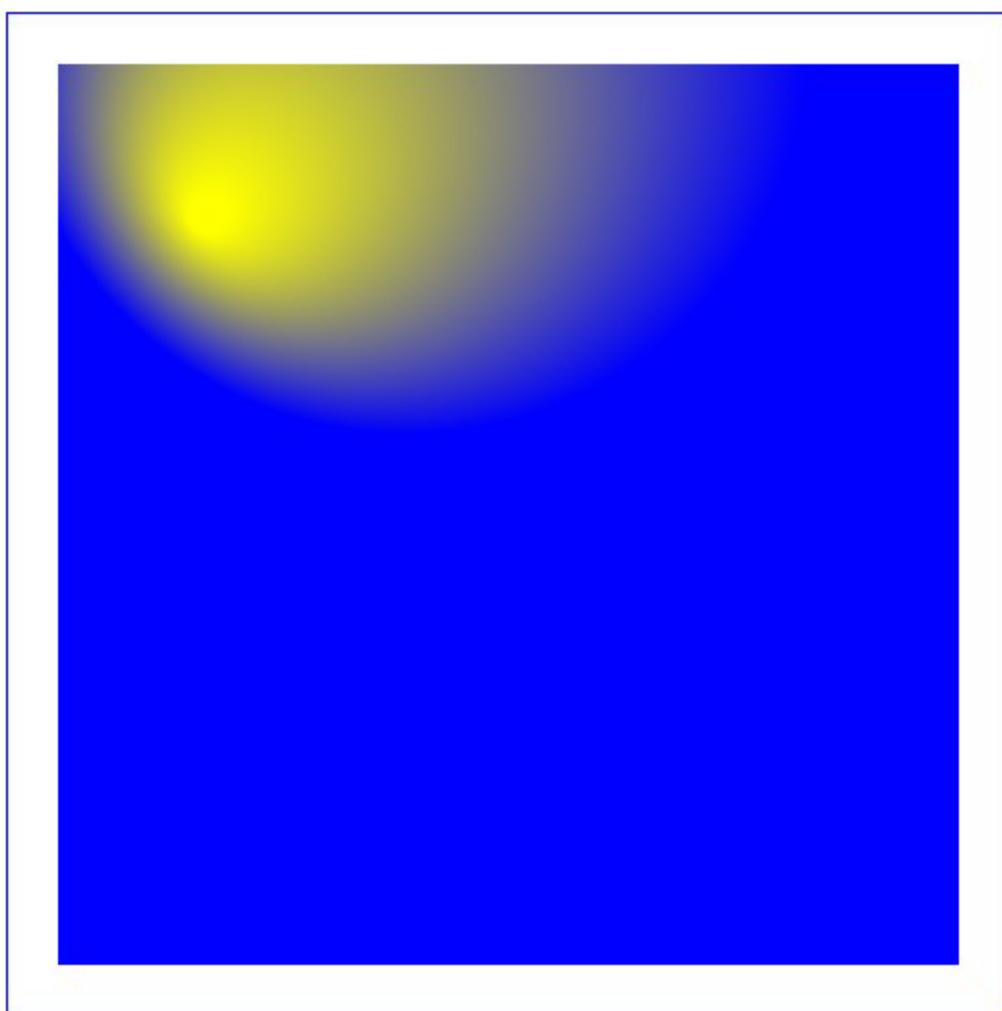
`r2` De straal van de tweede cirkel.

Met `x1, y1` bepaal je het middelpunt van de binnenste cirkel. De straal van die cirkel druk je met `r1` uit. Met `x2, y2` zet je het middelpunt van de buitenste cirkel uit, terwijl `r2` voor de straal van die tweede cirkel staat.

Voorbeeld radiaal

Met een radiaal verloop kan je diverse cirkelvormige verlopen in mekaar knutselen.

```
var verloop; //variabele declareren  
verloop = ctx.createRadialGradient(100,100,10,200,10,200); //  
verloop-object aanmaken  
verloop.addColorStop(0,"red"); //beginkleur  
verloop.addColorStop(1,"green"); //eindkleur  
ctx.fillStyle = verloop; //vulkleur instellen met verloop  
ctx.fillRect(25,25,450,450); //gevuld vierkant tekenen
```



7.1.2_verloopRadiaal.js

De binnenste cirkel heeft de vulkleur geel meegekregen. De buitenste cirkel is blauw. In het bovenstaande voorbeeld is het binnenste middelpunt bewust heel dicht tegen de rand van de canvas aan gelegd. Op die manier valt het verloop een stuk van de canvas af.

7.1.3 Patroon

Syntax

Met een patroon kan je een foto als vulling gebruiken. Die foto kan je binnen een vorm éénmaal of meerdere keren toepassen.

```
ctx.createPattern(afbeelding, "repeat");
```

De methode `createPattern()` verwacht twee argumenten. Eerst geef je het pad en de naam van de afbeelding, daarna volgt de wijze waarop het patroon gecreëerd moet worden.

`repeat` De afbeelding wordt zowel horizontaal als verticaal hernomen.

`repeat-x` Het beeld herhaalt enkel in de horizontale richting.

`repeat-y` Het beeld herhaalt enkel op de verticale as.

I joined MITAS because
I wanted **real responsibility**

The Graduate Programme
for Engineers and Geoscientists
www.discovermitas.com

Month 16

I was a construction supervisor in the North Sea advising and helping foremen solve problems

Real work
International opportunities
Three work placements

MAERSK

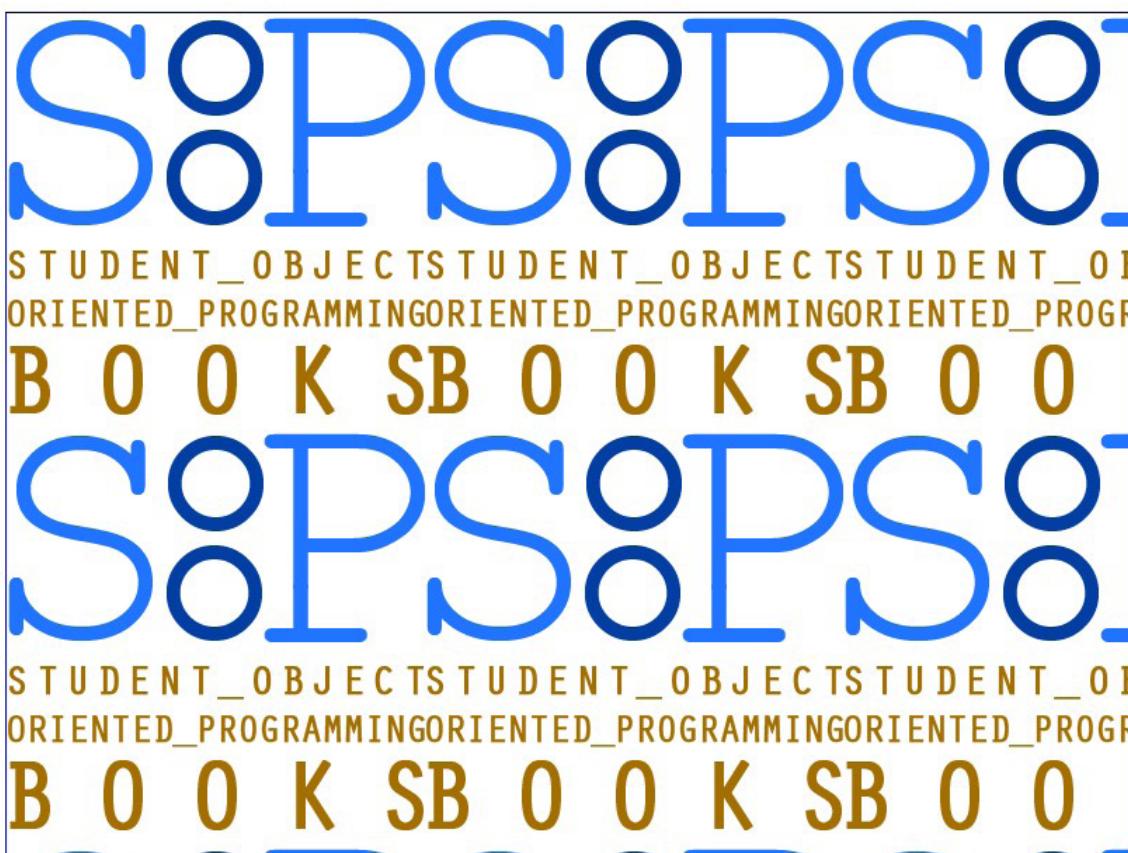
Download free eBooks at bookboon.com

Voorbeeld

Met een afbeelding kan je een zichzelf herhalend patroon aanmaken. Daarvoor staat de `createPattern()`-methode voor je klaar. Een voorbeeld.

```
var afbeelding = new Image();
afbeelding.src = "SoopbooksLogo.png";
afbeelding.onload = function() {

    var patroon; //variabele voor object patroon declareren
    patroon = ctx.createPattern(afbeelding,"repeat");
    ctx.rect(0,0,800,800); //vierkant om te vullen
    ctx.fillStyle = patroon; //patroon aan vulstijl koppelen
    ctx.fill(); //vierkant met vulstijl tekenen
}
```



7.1.3_patroon.js

Laten we de bovenstaande codes even bekijken. Eerst beginnen we met het laden van de afbeelding in een JavaScript-object.

```
var afbeelding = new Image();
afbeelding.src = "SoopbooksLogo.png";
```

Daarvoor declareer je eerst het object `afbeelding` als een `Image`-object. Voeg in de eigenschap `src` de pad en de naam van de afbeelding toe. Op dat moment staat de afbeelding bijna klaar in de variabele `afbeelding`. Bijna tenminste, bij een grote bestand kan dat laden naar het geheugen even duren. Daarom voeren we de code voor het patroon pas uit wanneer JavaScript de afbeelding volledig in de variabele `afbeelding` heeft gestopt.

```
afbeelding.onload = function() { }
```

Daarvoor zorgt de bovenstaande anonieme functie `function()`. De browser wacht met de uitvoering ervan totdat de afbeelding volledig geladen is (`onload`).

```
var patroon; //variabele declareren
```

Dan begin je met de code voor je patroon. Eerst declareer je een variabele. Die zal als object het patroon onderdak bieden. Vervolgens vul je dat object aan de hand van de methode `createPattern()` en bepaal je het soort herhaling.

```
patroon = ctx.createPattern(afbeelding, "repeat");
```

Daarna teken je een figuur waar je het patroon op wil toepassen.

```
ctx.rect(0,0,200,200); //vierkant om te vullen
```

Tot slot ken je dat patroon aan de vulstijl toe en geef je met de `fill()`-methode de vorm en het patroon weer.

```
ctx.fillStyle = patroon; //patroon aan vulstijl koppelen
ctx.fill(); //vierkant met vulstijl tekenen
```

7.2 Omlijnen

Syntax

Soms wil je een vorm niet vullen, maar wel omlijken. Dan pas je `stroke()` toe. Daarbij springt de eigenschap `strokeStyle` in het oog.

```
ctx.strokeStyle = "kleurwaarde";
```

Voorbeeld

```
ctx.beginPath(); //subpath instellen
ctx.strokeStyle = "red"; //lijnkleur instellen
ctx.lineWidth = 100; //lijndikte instellen
ctx.moveTo(100,60); //beginpunt lijn
ctx.lineTo(100,250); //eindpunt lijn
ctx.stroke(); //eerste lijn tekenen
ctx.closePath(); //subpath afsluiten
ctx.beginPath(); //subpath instellen
ctx.strokeStyle = "blue"; //lijnkleur instellen
ctx.lineWidth = 100; //lijndikte instellen
ctx.moveTo(180,30); //beginpunt lijn
```



Transform
the forces of wind

Join Vestas and innovate wind technology

Vestas

At Vestas we offer great opportunities to be part of a challenging and innovative work environment in a global company with more than 20,000 dedicated colleagues. Our state-of-the-art wind energy technology is part of the solution to reduce CO₂ emissions dramatically on a global scale and to make the world sustainable for future generations. Read more and apply at vestas.com/jobs

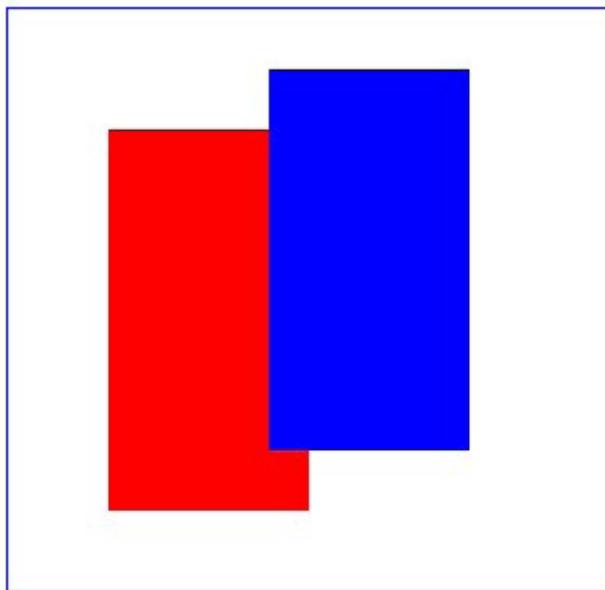
Wind. It means the world to us.TM

Download free eBooks at bookboon.com

57

Click on the ad to read more

```
ctx.lineTo(180,220); //eindpunt lijn  
ctx.stroke(); //tweede lijn tekenen  
ctx.closePath(); //subpath afsluiten
```



7.2_omlijnen.js

Met de context-eigenschap `strokeStyle` geef je een kleur aan de omlijning. Dat kan net zoals bij de `fillStyle` een kleurnaam, een hexadecimale waarde of een `rgb`-waarde zijn. In het bovenstaande voorbeeld zijn beide lijnen in een subpath ondergebracht.

Met de `stroke()`-methode geef je tot slot de omlijning effectief in de canvas weer.

```
ctx.stroke();
```

Kleurgebruik

Even een tip tussendoor. Merk je in het bovenstaande screenshot op dat wanneer een rood en een blauw vlak elkaar raken, je een diepte-effect krijgt?

8 Lijnen

Om illustraties te maken moet je figuren leren tekenen. Daarom kijken we nu eerst naar de meest eenvoudige figuur, namelijk de lijn. Dat is een rechte verbinding tussen twee punten. Echter bestaan er ook lijnen die meerderen punten verbinden, zoals een boog. De context voorziet dan ook heel wat soorten lijnen. We overlopen ze één voor één.

8.1 Rechte lijn

Syntax

Een rechte lijn vormt de meest eenvoudige lijn om te tekenen. Je stelt het beginpunt in, gevolgd door het eindpunt. Daarna geef je de instructie om de lijn tussen die twee punten zichtbaar te maken.

```
ctx.moveTo(x, y);  
ctx.lineTo(x, y);  
ctx.stroke();
```

In `moveTo()` zet je met `x` en `y` de coördinaten voor het beginpunt uit. Bij `lineTo()` doe je hetzelfde voor het eindpunt van de lijn. Wil je twee aangrenzende lijnen tekenen? Dan vormt het eindpunt van de vorige lijn, het beginpunt van de volgende lijn.

```
ctx.moveTo(x, y);  
ctx.lineTo(x1, y1);  
ctx.lineTo(x2, y2);  
ctx.stroke();
```

`x, y` beginpunt eerste lijn

`x1, y1` eindpunt eerste lijn en beginpunt tweede lijn

`x2, y2` eindpunt tweede lijn

Voorbeeld

Eerst bepaal je het beginpunt van de lijn. Staat je pen nog nergens klaar op de canvas, dan zal je dat met de methode `moveTo()` kunnen doen. Je kan die `moveTo()` overigens ook gebruiken wanneer je pen op de verkeerde plaats op de canvas gepositioneerd staat.

```
ctx.moveTo(100,50); //beginpunt pixel 100 op x-as en pixel 50
                  //op y-as
ctx.lineTo(100,250); //eindpunt pixel 100 op x-as en pixel 250
                     //op y-as
ctx.lineWidth = 10; //breedte van de lijn
ctx.stroke(); //effectief tekenen van de lijn
```



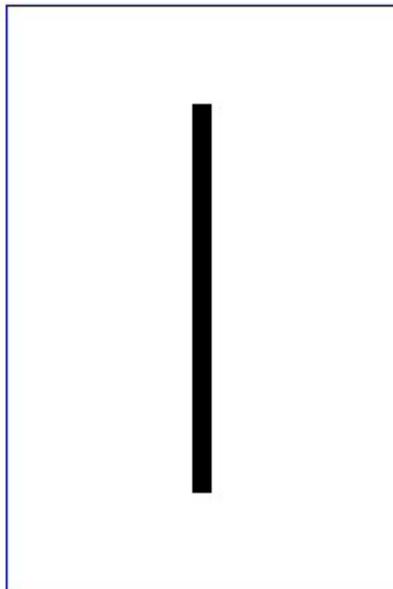
As a leading technology company in the field of geophysical science, PGS can offer exciting opportunities in offshore seismic exploration.

We are looking for new BSc, MSc and PhD graduates with Geoscience, engineering and other numerate backgrounds to join us.

To learn more our career opportunities, please visit www.pgs.com/careers



Download free eBooks at bookboon.com

**8.1_rechteLijn.js**

Met die `moveTo()` plaats je het beginpunt op de x-as (100) en op de y-as (50). Vervolgens geef je de coördinaten van het eindpunt op met de methode `lineTo`. Ook daar horen tussen de ronde haakjes de coördinaten thuis, eerst voor de x-as, gevolgd door de y-as. Met de opdracht `stroke()` trekt JavaScript vervolgens een rechte lijn tussen die twee punten.

Hoofdlettergevoelig

We drukken er nog eens op. Kijk goed naar de schrijfwijze van `moveTo()` met kapitaal T. Vergeet je die hoofdletter, dan zal je code niet werken, JavaScript is namelijk hoofdlettergevoelig.

Die spelling is doorheen heel JavaScript dus heel erg belangrijk, daarom blijven we erover zeuren. Voer je je code uit en er gebeurt niets, kijk dan je kapitalen in de namen van functies, methodes en variabelen alvast eens na.

8.1.1 Eigenschappen

Net zoals elke andere vorm bezit een lijn een eigen reeks aan eigenschappen. Zo kan je een lijn een dikte meegeven, een kleur en zelfs een hoekpunt.

Dikte `lineWidth`

```
ctx.lineWidth = 5;
```

Je bepaalt de dikte van een lijn met een getal. Dat getal drukt de breedte in een aantal pixels uit. 5 betekent hier dus een lijn van 5 pixels breed. Zo'n getal kan niet negatief zijn.

Merk daarbij op dat je het getal zonder aanhalingstekens plaatst.

Kleur `strokeStyle`

```
ctx.strokeStyle = "#000000";
```

Voor de kleur zou je een eigenschap met een naam als `lineColor` verwachten, niets is echter minder waar. Die eigenschap draagt namelijk de ondertussen bekende naam `strokeStyle`. Die eigenschap pas je dus niet alleen bij tekst toe, ook bij vormen.

Als waarde kan je kiezen. Ofwel geef je een hexadecimale kleurcode op, zoals `#000000` (zwart). We herhalen even. Ofwel gebruik je één van de vele voor gedefinieerde kleurennamen, zoals `black`, `blue`, `orange`, `yellow`, ... Tot slot kan je ook met een `rgb`-waarde werken, zoals `rgb(0, 255, 0)` voor groen.

Schrijf de waarde steeds tussen aanhalingstekens.

```
ctx.strokeStyle = "red";  
ctx.strokeStyle = "rgb(0,255,255)";
```

Uiteinde `lineCap`

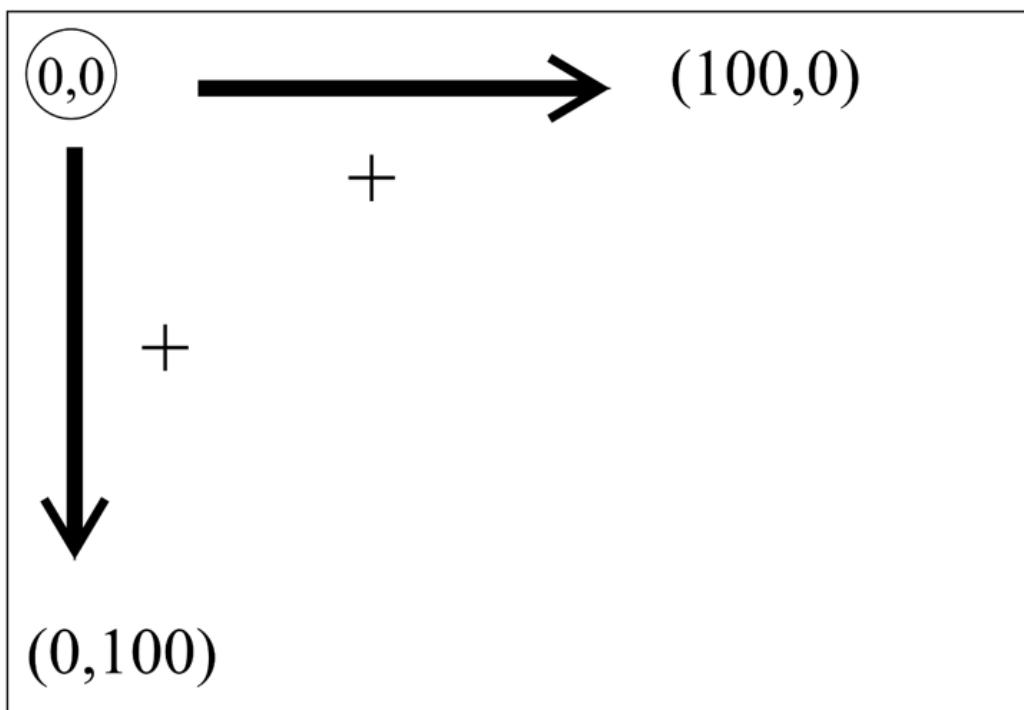
```
ctx.lineCap = "round";
```

Een `lineCap` bepaalt het uitzicht van de uiteinden van je lijn. Daarbij kan je afronden (`round`), standaard (`butt`) en verlengen (`square`). Bij die laatste is de lijn dan wat langer ten opzichte van een `butt`.

En ja hoor, de waarde plaats je steeds tussen aanhalingstekens.

8.1.2 Coördinaten

Maar hoe zit dat nu met die coördinaten precies in elkaar? Bekijk even de onderstaande illustratie.



Illustratie 2 – Coördinaten bij een computer. [\[soopbooks.com\]](http://soopbooks.com)



De linkerbovenhoek van je canvas is coördinaat $(0, 0)$. Het eerste getal verwijst naar de x-as, terwijl het tweede cijfer steeds de y-as aanduidt. Ga je naar rechts, dan stijgt de x-as-coördinaat. Ga je naar onder, dan stijgt de y-as-coördinaat.

Omgekeerd kan je ook negatieve waarden gebruiken. Zo staat de positie $(-100, 0)$ netjes buiten het canvasbeeld. Dat is soms handig wanneer je een illustratie eerst wil opbouwen, om ze daarna in een animatie schuivend in het beeld te laten binnen komen.

8.2 Kromme

Een kromme is een lijn die meerdere hoekpunten bezit. De canvas kent daarbij diverse krommen, zoals de boog, de kwadratische boog, de bezier-curve en de `arcTo()`. We overlopen die vier krommen.

8.2.1 Boog

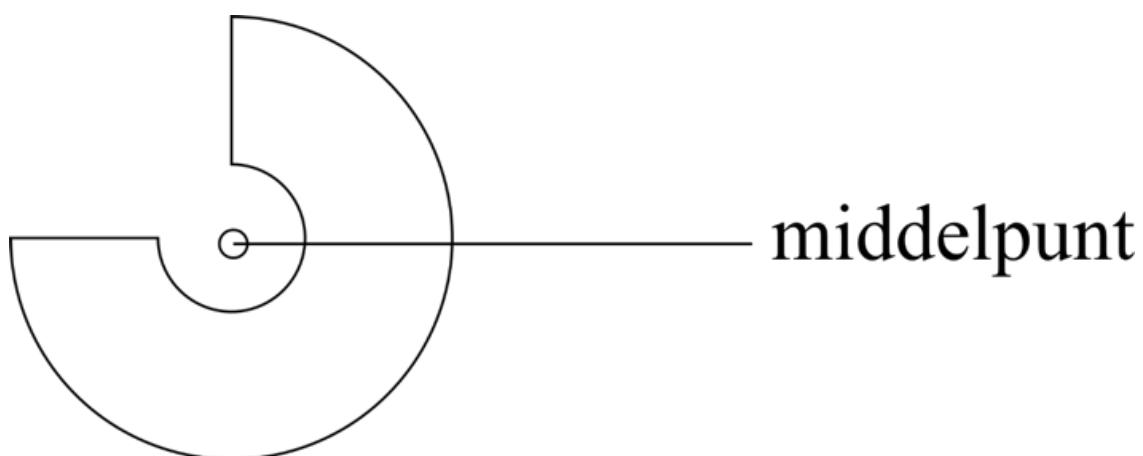
Syntax

De meest eenvoudige boog teken je met de methode `arc()`.

```
ctx.arc(x, y, straal, beginhoek, eindhoek, richting);
```

De `arc()`-methode vraagt maar liefst om zes verschillende parameters.

- x Om een boog te tekenen, schets je eigenlijk een cirkel. Alleen toon je niet de volledige cirkel. Daardoor staat x voor het x-coördinaat van het middelpunt van een (denkbeeldige) cirkel.
- y Met y vervolledig je het coördinaat van het middelpunt van de (denkbeeldige) cirkel.

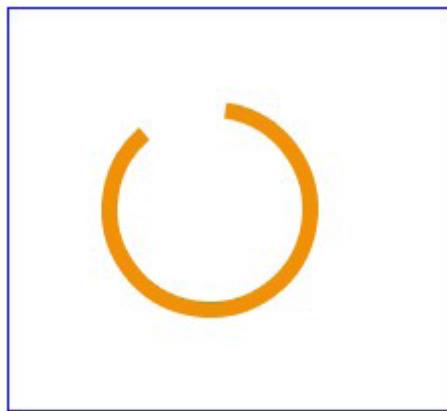


Illustratie 3 – Je kan de boog zo tot een cirkel vervolledigen. [soopbooks.com]

- straal De straal bepaalt bij een cirkel hoe breed die cirkel vanuit het middelpunt is. Je boog groeit dus met de straal mee.
- beginhoek Welke beginhoek vormt de boog ten opzichte van het middelpunt?
- eindhoek Welke eindhoek vormt de boog ten opzichte van het middelpunt?
- richting De boog wordt met de wijzers van de klok mee getekend. Wil je dat anders? Geef dan true als waarde op. Standaard – je hoeft dat zelfs niet te vermelden – staat de waarde namelijk op false.

Voorbeeld

```
ctx.arc(100, 100, 50, 30, 4, false); //boogvorm bepalen
ctx.strokeStyle = "#F09108"; //oranje lijnkleur
ctx.lineWidth = 8; //lijnbreedte
ctx.stroke(); //boog weergeven
```



8.2.1_boog.js

Het bovenstaande voorbeeld toont een oranje boog. Die boog is bijna gesloten tot een cirkel. Het middenpunt van de denkbeeldige cirkel staat op coördinaten 100, 100. De straal bedraagt 50 pixels. De beginhoek staat op 30 graden, terwijl de eindhoek op 4 graden afklokt.

De `arc()`-methode zou ook korter geschreven kunnen worden, je mag de richting – die nu op de standaardwaarde `false` staat – ook weglaten.

```
ctx.arc(100, 100, 50, 30, 4); //zonder richting
```

Eigenschappen

Een boog is een kromme lijn. Daarom kan je alle eigenschappen van de lijn ook op een boog toepassen (`lineWidth`, `strokeStyle` en `lineCap`). Zie bij de eigenschappen van de lijn.

8.2.2 Bézier-kromme

Syntax

Een bázier-kromme is een boog die uit meerdere punten bestaat. De boog vertrekt bij het eerste punt, gaat vervolgens naar het tweede punt, daarna naar het derde punt, ..., steeds in de richting van het laatste punt. Daarbij zorgen twee controlepunten voor de eigenlijke kromming. Die moet je binnen de methode `bezierCurveTo()` eerst opgeven, daarna duid je de x- en y-waarden van het eindpunt aan.

```
ctx.bezierCurveTo(xcp1, ycp1, xcp2, ycp2, x, y);
```

`xcp1` De x-coördinaat van het eerste controlepunt.

`ycp1` De y-coördinaat van het eerste controlepunt.

`xcp2` De x-coördinaat van het tweede controlepunt.



`ycp2` De y-coördinaat van het tweede controlepunt.

`x` De x-coördinaat van het middelpunt.

`y` De y-coördinaat van het middelpunt.

Voorbeeld

```
ctx.beginPath(); //start een subpad
ctx.lineWidth = 6; //breedte van de lijn
ctx.strokeStyle = "rgb(125,30,69)"; //paarse lijnkleur
ctx.moveTo(30,30); //plaats het beginpunt
ctx.bezierCurveTo(30,110,100,110,100,30); //bepaal de vorm van
                                            //de boog
ctx.stroke(); //maak de boog zichtbaar op de canvas
ctx.closePath(); //sluit het subpad af
```



8.2.2_bezier.js

Begin met een pad. Plaats daarna je tekenpen op de coördinaten `30, 30` met de methode `moveTo()`. Dat is het uitgangspunt van je boog. Vervolgens zet je het eerste controlepunt op coördinaat `30, 110` uit. Het tweede controlepunt valt op `100, 110` post. Het eindpunt van de boog heeft tot slot `100, 30` als thuishaven.

Eigenschappen

Een bázier-kromme is een kromme lijn. Daarom kan je alle eigenschappen van de lijn ook op een boog toepassen (`lineWidth`, `strokeStyle` en `lineCap`). Zie bij de eigenschappen van de lijn.

8.2.3 Kwadratische boog

Syntax

De kwadratische boog is een variant op de bézier-kromme. Terwijl de gewone bézier-kromme twee controlepunten bezit, beschikt de kwadratische boog over slechts één controlepunt.

```
ctx.quadraticCurveTo(xcp, ycp, x, y);
```

xcp De x-coördinaat van het controlepunt.

ycp De y-coördinaat van het controlepunt.

x De x-coördinaat van het middelpunt.

y De y-coördinaat van het middelpunt.

Voorbeeld

```
ctx.beginPath(); //subpath beginnen
ctx.lineWidth = 8; //lijnbreedte instellen
ctx.strokeStyle = "green"; //groene lijnkleur
ctx.moveTo(30,30); //pen klaarzetten
ctx.quadraticCurveTo(30,110,100,30); //kwadratische boog
//omschrijven
ctx.stroke(); //boog tekenen
ctx.closePath(); //subpath afsluiten
```



8.2.3_kwadratisch.js

Bij een kwadratische boog begin je ook met een pad. Dan stel je een beginpunt in en bepaal je die boog door de x- en y-coördinaten van het controlepunt in te stellen en de x- en y-coördinaten van het middelpunt op te geven. Met de `stroke()` geef je de boog effectief weer. Vergeet niet om het path af te sluiten.

Eigenschappen

Een kwadratische boog is een kromme lijn. Daarom kan je alle eigenschappen van de lijn ook op een boog toepassen (`lineWidth`, `strokeStyle` en `lineCap`). Zie daar.

8.2.4 arcTo

Syntax

De `arcTo`-methode lijkt tot slot sterk op `arc()`. Toch is er een belangrijk verschil. De `arcTo()` gebruikt twee controlepunten. Wat is dan het verschil met een bázier-kromme, denk je dan? Wel, een `arcTo()` kan je handig gebruiken om een hoek tussen twee rechte lijnen af te ronden.

```
ctx.arcTo(x1, y1, x2, y2, straal);
```

`x1` De x-coördinaat van het controlepunt.

`y1` De y-coördinaat van het controlepunt.

`x2` De x-coördinaat van het middelpunt.

`y2` De y-coördinaat van het middelpunt.

`straal` De straal van de denkbeeldige cirkel.

Success here is about overcoming challenges – especially the ones we didn't even think of. I look forward to that.

- Scott, Mechanical Engineering Group Leader

Produce More. Conserve More. Improve Lives.

Monsanto has always embraced innovation and always focused on helping to make a better world. You can see it in our groundbreaking products and in our dynamic environment where your skills and your career can grow and develop. We know that every day, new ideas can come from anyone, anywhere. At Monsanto, you'll be respected, you'll contribute to the bottom line and you'll help farmers feed the world.

Start right now: www.monsanto.com/students

Monsanto is an equal opportunity employer, we value a diverse combination of ideas, perspectives and cultures.
EEO/AA EMPLOYER M/F/D/V © 2013 Monsanto Company

MONSANTO

Voorbeeld

```
ctx.beginPath(); //subpath beginnen  
ctx.moveTo(30,30); //beginpunt  
ctx.lineTo(100,30); //rechte lijn naar rechts  
ctx.arcTo(140,110,140,30,5); //afgeronde hoek  
ctx.lineTo(140,130); //rechte lijn naar onder  
ctx.stroke(); //tekenen  
ctx.closePath(); //subpath afsluiten
```



8.2.4_arcTo.js

Begin met een subpad. Daarna zet je het beginpunt uit met `moveTo (30, 30)`. Dan teken je twee lijnen die loodrecht op elkaar staan. De hoek ertussen rond je met de `arcTo ()` af. De `stroke ()`-methode maakt je tekenwerk op de canvas zichtbaar. Sluit het subpath tot slot af.

Eigenschappen

Een boog met twee controlepunten is een kromme lijn. Daarom kan je alle eigenschappen van de lijn ook op een boog toepassen (`lineWidth`, `strokeStyle` en `lineCap`). Zie bij de eigenschappen van de lijn.

9 Joins van lijnen

Wanneer je twee lijnen na elkaar tekent, zijn die niet noodzakelijk netjes met elkaar verbonden. Dan pas je de `lineJoin`-eigenschap toe. Die eigenschap bepaalt het soort hoek dat twee lijnen met elkaar maken. Dat kan recht, spits of rond zijn.

Syntax

```
ctx.lineJoin = "waarde";
```

De eigenschap `lineJoin` kan drie waarden aannemen, `miter`, `round` en `bevel`.

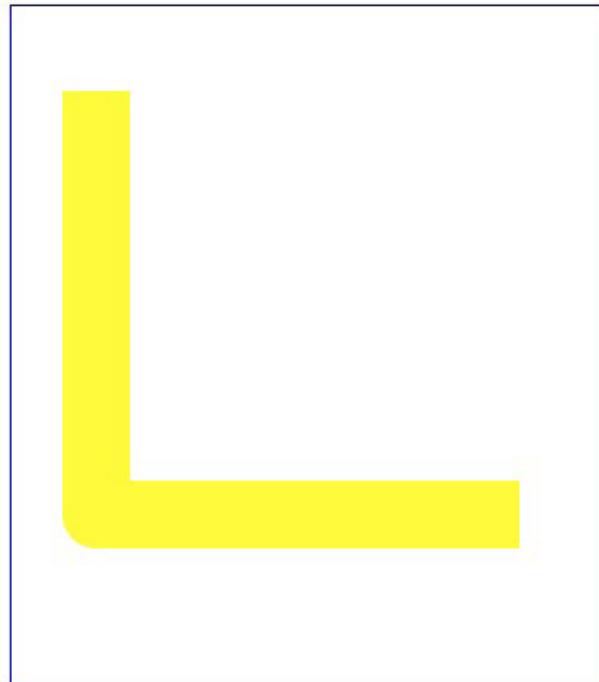
`miter` Geef je geen specifieke waarde voor `lineJoin` op, dan vormt `miter` de standaardwaarde. Daarmee maak je een gewone klassieke rechte hoek.

`round` `round` zorgt ervoor dat de hoek afgerond wordt.

`bevel` De waarde `bevel` zorgt voor een spitse hoek.

Voorbeeld

```
ctx.beginPath(); //begin een nieuw subpad
ctx.lineWidth = 40; //breedte van de lijn
ctx.strokeStyle = "rgb(255,250,60)"; //kleur van de lijn
ctx.moveTo(50,50); //zet de pen klaar op punt 50, 50
ctx.lineTo(50,300); //trek vanuit 50,50 een lijn naar punt
                    //50,300
ctx.lineTo(300,300); //trek vanuit 50,300 een lijn naar punt
                    //300,300
ctx.lineJoin = "round"; //verbond de punten van de twee
                        //bovenstaande lijnen met een afgeronde
                        //hoek
ctx.stroke(); //tekenen van de lijn
ctx.closePath(); //subpad afsluiten
```

**9**_arcTo.js

Begin met een nieuw subpad. Daarna teken je twee lijnen. Met de eigenschap `lineJoin` bepaal je tot slot de hoek. In het voorbeeld is dat een afgeronde hoek.



10 Rechthoeken

Met lijnen kan je heel wat rechthoekfiguren tekenen. Toch is het handiger om een rechthoek met één enkele methode te ontwerpen. Je hebt daarbij de keuze uit drie opties, de gewone rechthoek, de gevulde rechthoek en de omlijnde rechthoek. Daarnaast kan je ook uitsnijdingen uit een rechthoek realiseren.

10.1 Rechthoek

Syntax

Met de methode `rect()` ontwerp je een rechthoek. De methode verwacht de input van vier parameters. Met de argumenten `x` en `y` bepaal je de linkerbovenhoek van de rechthoek. Vervolgens vertel je de `breedte` en de `hoogte`. Op basis van die laatste argumenten zijn namelijk alle gegevens bekend om de rechthoek te vervolledigen.

```
ctx.rect(x, y, breedte, hoogte);
```

`x` De x-coördinaat van de linkerbovenhoek van de rechthoek.

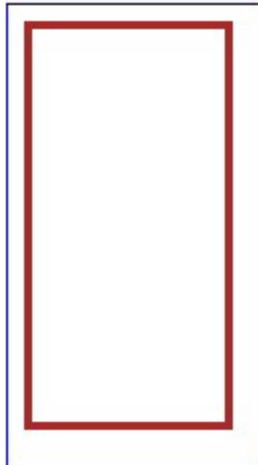
`y` De y-coördinaat van de linkerbovenhoek van de rechthoek.

`breedte` De breedte van de rechthoek.

`hoogte` De hoogte van de rechthoek.

Voorbeeld

```
ctx.rect(10,10,100,200); //rechthoek omschrijven  
ctx.strokeStyle = "brown"; //bruine lijnkleur  
ctx.lineWidth = 4; //lijnbreedte  
ctx.stroke(); //rechthoek tekenen
```

**10_rechthoek.js**

Bepaal eerst de positie en de grootte van de rechthoek. Daarna stel je een vulkleur en een lijnbreedte in. Vergeet echter niet om een `stroke()` of een `fill()` aan te roepen om de rechthoek effectief te tonen. De `rect()`-methode toont het resultaat namelijk niet op het scherm.

Ook kan je een gevulde variant op de bovenstaande code maken met `fillStyle` en `fill()` in plaats van `strokeStyle` en `stroke()`.

**We ask you
WHERE DO YOU
WANT TO BE?**

TOMTOM 

TomTom is a place for people who see solutions when faced with problems, who have the energy to drive our technology, innovation, growth along with goal achievement. We make it easy for people to make smarter decisions to keep moving towards their goals. If you share our passion - this could be the place for you.

Founded in 1991 and headquartered in Amsterdam, we have 3,600 employees worldwide and sell our products in over 35 countries.

For further information, please visit tomtom.jobs



10.2 Varianten

Bij de gewone `rect()`-methode moet je de `stroke()` of de `fill()` gebruiken om de rechthoek te tonen op het scherm. Toch bestaan er ook varianten die de rechthoek in één beweging omschrijven én tekenen.

10.2.1 `fillRect`

Syntax

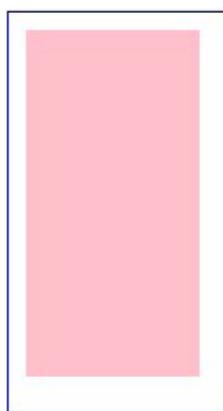
```
ctx.fillRect(x1, y1, breedte, hoogte);
```

x1	De x-coördinaat van de linkerbovenhoek.
y1	De y-coördinaat van de linkerbovenhoek.
breedte	De breedte van de rechthoek.
hoogte	De hoogte van de rechthoek.

Voorbeeld

Bij een `fillRect()` maak je een rechthoek die meteen de vulkleur uit de context-eigenschap `fillStyle` toepast. Je hoeft dan ook geen methode `fill()` aan te roepen. Die taak neemt de `fillRect()`-methode direct voor zijn rekening.

```
ctx.fillStyle = "pink"; //roze vulkleur  
ctx.fillRect(10,10,100,200); //rechthoek omschrijven én tekenen
```



10.2.1_fillRect.js

Met slechts twee lijnen code is er nu een roze rechthoek te zien.

10.2.2 strokeRect()

Syntax

```
ctx.strokeRect(x1,y1,breedte,hoogte);
```

x1 De x-coördinaat van de linkerbovenhoek.

y1 De y-coördinaat van de linkerbovenhoek.

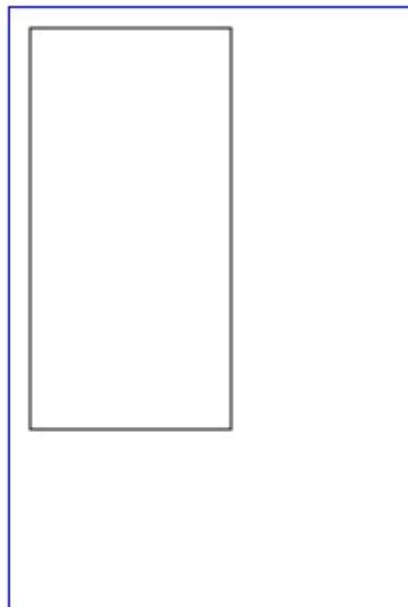
breedte De breedte van de rechthoek.

hoogte De hoogte van de rechthoek.

Voorbeeld

Met `strokeRect()` maak je een rechthoek die meteen weergegeven wordt. Je hoeft de `stroke()`-methode niet meer aan te roepen.

```
ctx.strokeRect(10,10,100,200); //rechthoek omschrijven én tonen
```



10.2.2_strokeRect.js

Eén lijn code volstaat om een eenvoudige rechthoek te tonen.

10.3 Uitsnijden

Syntax

Uit een bestaande rechthoek kan je een zone uitsnijden. Dat doe je met de methode `clearRect()`. Eerst bepaal je met `fillStyle()` de vulkleur. Daarna teken je een gevulde rechthoek. Niets nieuws onder de zon. Dan bepaal je met `clearRect()` welke zone je wil uitsnijden. Die zone vormt een nieuwe rechthoek, met de x- en y-coördinaat van de linkerbovenhoek, de breedte en de hoogte.

```
ctx.clearRect(x1, y1, breedte, hoogte);
```

x1	De x-coördinaat van de linkerbovenhoek.
y1	De y-coördinaat van de linkerbovenhoek.
breedte	De breedte van de rechthoek.
hoogte	De hoogte van de rechthoek.

Melvin Spalburg
Project Engineering Manager

AkzoNobel

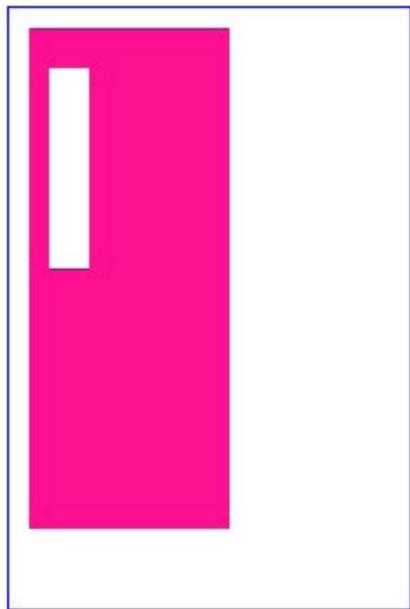
Improve
that plant's
PERFORMANCE
together

To find out more about Melvin,
please visit our website
www.akzonobel.nl/careers

07251_190813

Voorbeeld

```
ctx.fillStyle = "#FF1090"; //kies een vulkleur  
ctx.fillRect(10,10,100,250); //teken een rechthoek  
ctx.clearRect(20,30,20,100); //snij een kolom uit
```



10.3_uitsnijden.js

Start met een vulkleur. Daarmee teken je een rechthoek. Vervolgens omschrijf je een nieuwe rechthoek die automatisch wit als vulkleur mee krijgt.

Je kan de `clearRect` als een gom beschouwen. Daardoor kan je die methode ook gebruiken om je canvas volledig leeg te maken. Dat is nuttig wanneer je animaties maakt.

```
ctx.clearRect(0,0,ctx.canvas.width,ctx.canvas.height);
```

Met de eigenschappen `width` en `height` kan je de breedte en de hoogte van de canvas bepalen.

`ctx.canvas.width` De breedte van de canvas.

`ctx.canvas.height` De hoogte van de canvas.

11 Cirkels

Naast rechthoeken kan je ook cirkels tekenen. Daarbij kan je kiezen uit volledige cirkels, of uit stukken van cirkels. We zetten die mogelijkheden onder elkaar.

11.1 Volle cirkel

Om een cirkel te tekenen, kan je met `arc()` werken.

Syntax

Een cirkel tover je met de context-methode `arc()` op het scherm. Daarbij heeft de methode een resem aan argumenten nodig.

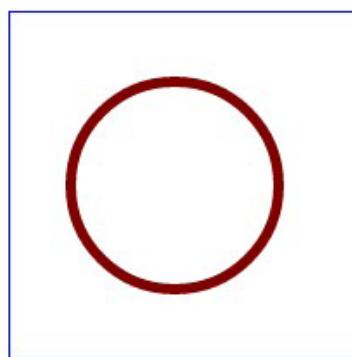
```
ctx.arc(x,y,r,begin,eind,richting);
```

We overlopen de argumenten één voor één.

<code>x</code>	De x-coördinaat van het middelpunt van de cirkel.
<code>y</code>	De y-coördinaat van het middelpunt van de cirkel.
<code>r</code>	De straal van de cirkel.
<code>begin</code>	De beginhoek.
<code>eind</code>	De eindhoek.
<code>richting</code>	De tekenrichting tegen de wijzers van de klok.

Voorbeeld

```
ctx.strokeStyle = "maroon"; //kleur van de lijn
ctx.lineWidth = 6; //breedte van de lijn
ctx.arc(95,100,60,0,Math.PI*2,false); //boog tekenen
ctx.stroke(); //boog tonen;
```



11.1.1_arc.js

Het bovenstaande voorbeeld tekent een volledige cirkel. Daarvoor neem je starthoek 0 en eindhoek `Math.PI*2`. De pi-waarde zit namelijk in het `Math`-object van JavaScript vervat. De waarde `false` betekent dat je de cirkel met de wijzers van de klok mee tekent. Kies je voor `true`, dan teken je in de andere richting. Die richting is bij een volle cirkel niet van belang, enkel wanneer je een stuk van een cirkel tekenst.

Met de `stroke()`-methode teken je de cirkel effectief op de canvas.

11.2 Deel cirkel

Als je weet dat je met `Math.PI*2` een volledige cirkel tekent, dan kan je op basis van die kennis met de `arc()`-methode ook een stuk van een cirkel opbouwen.



TNO innovation
for life

EARTH, LIFE AND SOCIAL SCIENCES

INNOVATIES DIE WERKEN

Mensen, organisaties en onze samenleving bepalen uiteindelijk welke innovaties voor hen van groot belang zijn en op welke wijze ze daarmee kunnen en willen werken. Daarvoor heb je innovaties nodig, maar ook veel kennis van menselijk gedrag - zowel individueel als in teams - van organisaties, maar ook van de stuwende kracht achter maatschappelijke processen. Die kennis levert het expertisegebied Earth, Life and Social Sciences.

Zo'n 1300 mensen variërend van onder meer moleculair bioloog, chemicus, ICT'er, bedrijfskundige, wiskundige, psycholoog en aardwetenschapper zijn hierbij betrokken. Juist de synergie van creatieve combinaties van vakgebieden maakt het ons mogelijk om uitdagende problemen voor onze klanten aan te pakken en helpen op te lossen.

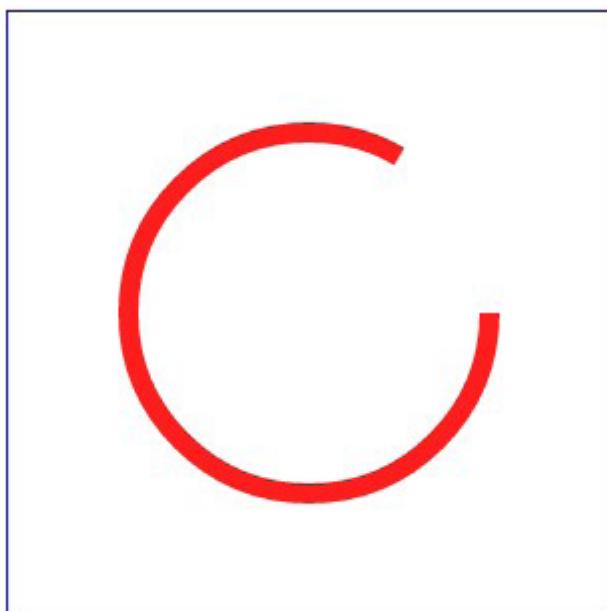
Syntax

```
ctx.arc(x,y,r,begin,eind,richting);
```

x	De x-coördinaat van het middelpunt van de cirkel.
y	De y-coördinaat van het middelpunt van de cirkel.
r	De straal van de cirkel.
begin	De beginhoek.
eind	De eindhoek.
richting	De tekenrichting tegen de wijzers van de klok.

Voorbeeld

```
ctx.strokeStyle = "rgb(255,30,30)"; //kleur van de lijn  
ctx.lineWidth = 10; //breedte van de lijn  
ctx.arc(50,100,20,0,Math.PI/0.6,false); //boog opbouwen  
ctx.stroke(); //boog tekenen
```



11.2_arc.js

Deel je pi door 0.6 – let op het punt in plaats van een komma – snij je bijvoorbeeld een stuk van de cirkel af.

12 Clipping

Soms bouw je overlappende figuren, waarvan je alleen de overlappende delen zichtbaar wil houden. Dan helpt clipping je meteen een stap vooruit. Je kan clipping met een masker vergelijken.

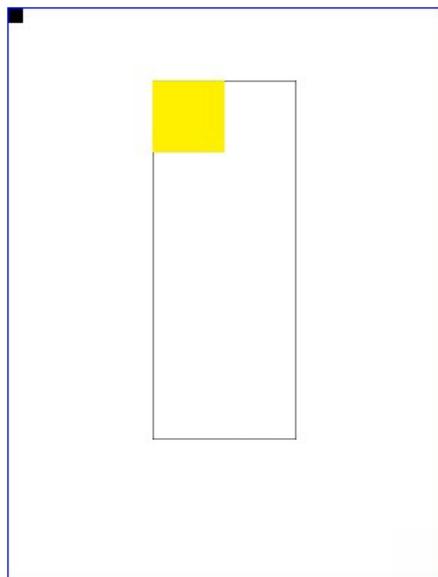
Syntax

```
ctx.clip();
```

De methode `clip()` bezit geen argumenten.

Voorbeeld

```
ctx.save(); //instellingen context bewaren
ctx.rect(100, 50, 100, 250); //basisrechthoek tekenen
ctx.clip(); //clipping actief zetten
ctx.stroke(); //basisrechthoek zichtbaar maken
ctx.fillStyle = "#FFF000"; //vulkleur instellen
ctx.fillRect(0, 0, 150, 100); //af te snijden rechthoek
                                //met vulkleur tekenen
ctx.restore(); //vorige context-instellingen terug zetten
                //met standaard vulkleur zwart
ctx.beginPath(); //nieuw subpad beginnen
ctx.fillRect(0,0,10,10); //gevulde rechthoek tekenen
ctx.closePath(); //subpath afsluiten
```

**12_clipping.js**

Je ziet hoe we een rechthoek tekenen, met daarbinnen nog twee andere rechthoeken. Je zou dat perfect met drie lagen kunnen ontwerpen, echter is het soms verstandig om dat als één geheel te benaderen, je haakt – of clipt – die drie vormen dan aan elkaar vast.

DENMARK IS HIRING



Are you looking to further your cleantech career in an innovative environment with excellent work/life balance? Think Denmark!
Visit cleantech.talentattractiondenmark.com

"In Denmark you can find great engineering jobs and develop yourself professionally. Especially in the wind sector you can learn from the best people in the industry and advance your career in a stable job market."

Mireia Marrè,
Advanced Engineer from Spain.
Working in the wind industry in Denmark since 2010.

Empower
your career
- think Denmark

Dat clippen doe je in enkele stappen. Eerst ga je de huidige context opslaan, daarna teken je het hoofdobject, vervolgens vertel je JavaScript dat je met de clip-functie aan de slag wil. Verder teken je de te koppelen objecten en sluit je af met het terugplaatsen van de vorige ctx.

We overlopen die bovenstaande stappen één voor één.

12.1 Bewaren

Je bent niet verplicht om bij clipping de huidige context-instellingen te bewaren, toch is het raadzaam. Meestal stel je aan het begin van je code de standaardkleuren, lettertypes en ander opmaakongeïn in, terwijl je bij een clipping vaak bijvoorbeeld andere kleuren begint toe te passen. Wanneer je je standaardopmaak dan even bewaart, roep je die na de clipping zo weer terug op.

```
ctx.save(); //instellingen context bewaren
```

12.2 Gastheer

Vervolgens bepaal je de zone waarbinnen je de overlappende delen van de andere vormen in zichtbaar wil toveren. Die zone noemen we de basiszone. Je zou zo'n basiszone ook als een canvas binnen een canvas kunnen beschouwen. Of nog, je creëert een nieuwe kleinere context binnen je oorspronkelijke ctx.

```
ctx.rect(100, 50, 100, 250); //basisrechthoek tekenen
ctx.stroke(); //basisrechthoek zichtbaar maken
```

Je merkt dat we een rechthoek met een `stroke()` tekenen. Echter kan je ook met een vulkleur werken, je figuren die binnen de basiszone vallen, vullen je basiszone namelijk niet noodzakelijk volledig op. De basiszone vormt dan ook een gastheer voor alle geclipte figuren.

12.3 Clip-functie

De `clip()`-functie roep je zonder parameters aan.

```
ctx.clip(); //clipping actief zetten
```

Ze zorgt ervoor dat al je verdere tekenwerk enkel binnen de gastheer getoond wordt. Trek je bijvoorbeeld een lijn met coördinaten buiten de basiszone, dan zal die lijn simpelweg niet te zien zijn.

```
ctx.moveTo(0,0);
ctx.lineTo(0,30);
ctx.stroke();
```

Wanneer je de bovenstaande lijnen aan de code zou toevoegen, dan zal je geen enkel verschil in de uitvoer merken.

12.4 Clip-figuren

Zodra de clipping geactiveerd is, zal elke figuur die je tekent aan de basiszone gelinkt worden.

```
ctx.fillStyle = "#FFF000"; //vulkleur instellen
ctx.fillRect(0, 0, 150, 100); //af te snijden rechthoek
                                //met vulkleur tekenen
```

In de bovenstaande lijnen zie je hoe er een vulkleur bepaald wordt. De standaardkleur zwart die in de bewaarde context zit (`ctx.save()`), zetten we nu naar een geelachtige tint om. Vervolgens tekenen we een rechthoek met de linkerbovenhoek helemaal linksboven in de canvas. De coördinaten die we opgeven blijven dus relatief ten opzichte van de canvas en niet ten opzichte van de basiszone.

Die basiszone begint pas op coördinaat 100, 50. Daardoor zal de ingekleurde geclipte rechthoek pas zichtbaar getoond worden vanaf dat coördinaat.

12.5 Terugkeren

Ben je klaar met het koppelen van objecten aan de basiszone? Wil je bijvoorbeeld een rechthoek tonen die wel buiten de basiszone getoond wordt? Haal de oorspronkelijke context dan terug op.

```
ctx.restore(); //vorige context-instellingen terug zetten
                //met standaard vulkleur zwart
```

Echter volstaat het niet om nu direct een nieuwe gevulde rechthoek te tekenen. Begin een nieuw pad, anders loop je het risico dat je geelachtige geclipte rechthoek de standaardkleur zwart overneemt.

```
ctx.beginPath(); //nieuw pad beginnen
ctx.fillRect(0,0,10,10); //gevulde rechthoek tekenen
```

13 Transformaties

Wanneer je zo nu en dan met fotobewerking aan de slag gaat, of illustraties in een tekenpakket opbouwt, dan klinkt het begrip transformatie je vast bekend in de oren. Zo kan je figuren vergroten, draaien of vervormen, of kan je een object binnen je werkruimte van plaats veranderen. De canvas heeft die truckjes ook allemaal in huis. Kijk je even mee?

13.1 Schalen

Syntax

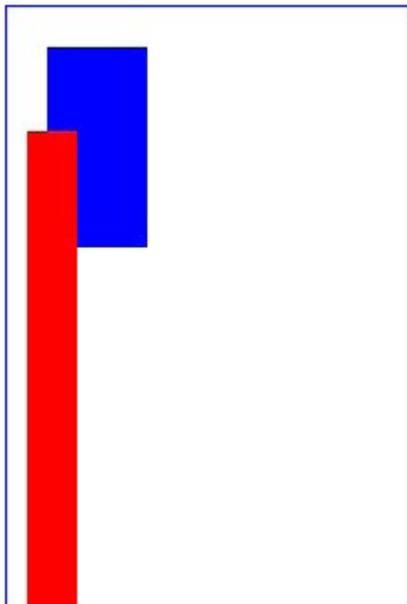
Wanneer je een figuur wil vergroten of verkleinen, dan staat de methode `scale()` voor je klaar. Daarbij kan je zowel de hoogte als de breedte apart manipuleren.

```
ctx.scale(breedte, hoogte);
```

Voorbeeld

Je tekent eerst een blauwe rechthoek. Vervolgens ga je diezelfde rechthoek – in het rood – in geschaalde vorm weergeven.

```
ctx.fillStyle = "blue"; //blauwe vulkleur
ctx.fillRect(20, 20, 50, 100); //gevulde rechthoek omschrijven
                           //én tekenen
ctx.scale(0.5, 3.1); //schaalwaarden instellen
ctx.fillStyle = "red"; //rode vulkleur
ctx.fillRect(20, 20, 50, 100); //gevulde rechthoek omschrijven
                           //én tekenen
```

**13.1_scale.js**



innovation + you

Some see lighting that captivates. **You see hospitals that comfort**

Imagine the impact you can have
Philips is a diversified technology company, focused on improving people's lives through meaningful innovations. As a world leader in Healthcare, Consumer Lifestyle and Lighting, Philips integrates technologies and design into people-centric solutions, based on fundamental customer insights.

Meaningful contribution
At Philips, we strive to make the world healthier and more sustainable through innovation. Our goal is to improve the lives of 3 billion people a year by 2025. To achieve this we are looking for passionate, insightful people that have the drive to deliver meaningful results together with a great team.

If you want to deliver a meaningful contribution and want to have an impact on the quality of people's lives, take your chance to start your future at Philips as an intern, a trainee or as a talent in a regular job.

PHILIPS

Find out more on www.philips.nl/careers



Click on the ad to read more

`scale()` verwacht twee argumenten. Het eerste argument drukt de verandering van de breedte uit, terwijl het tweede argument de hoogte wijzigt. Beide argumenten druk je in een percentage uit. Zo staat 1 voor 100 procent (geen wijziging), terwijl 0.5 voor 50 procent staat (een halvering). Let je nog steeds op het punt bij de decimale getallen in JavaScript? Je kan de breedte en de hoogte los van elkaar wijzigen.

```
ctx.scale(0.5, 3.1); //breedte halveren, hoogte met 310  
//procent verhogen
```

13.2 Roteren

Syntax

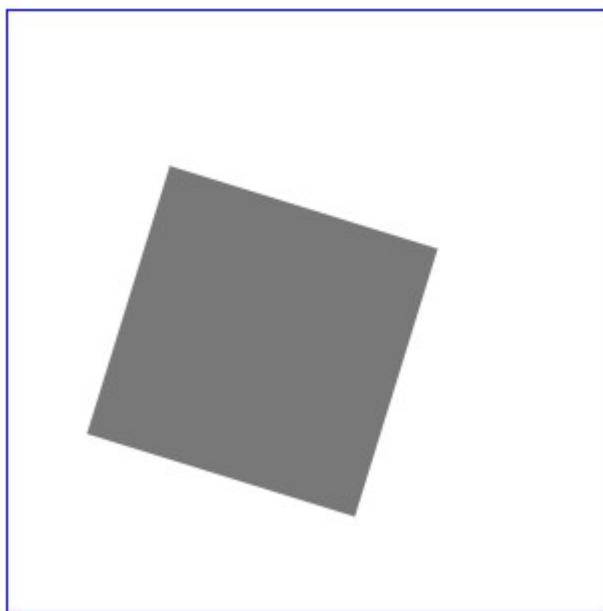
```
ctx.rotate(getal);
```

getal Een waarde tussen 0 en 1.

Voorbeeld

Vervolgens kan je een object zoals een rechthoek ook om de eigen as draaien.

```
ctx.fillStyle = "rgb(120,120,120)"; //vulkleur  
ctx.rotate(.3); //rotatiewaarde  
ctx.fillRect(100, 50, 140, 140); //gevulde rechthoek  
//omschrijven én tekenen
```



13.2_rotate.js

Eerst bepaal je een grijstint als vulkleur. Daarna pas je de `rotate()`-functie toe. Zoals je merkt schreven we hier `.3` in plaats van `0.3`. Op die manier wordt je code langs de ene kant wat beknopter, langs de andere kant is ze iets minder leesbaar. Aan jou de keuze.

13.3 Vervormen

Syntax

Met de `transform()`-methode, kan je figuren vervormen.

```
ctx.transform(hs, hh, vh, vs, hv, vv);
```

`hs` Horizontaal schalen uitgedrukt in een percentage.

`hh` Horizontale helling uitgedrukt in een percentage.

`vh` Verticale helling uitgedrukt in een percentage.

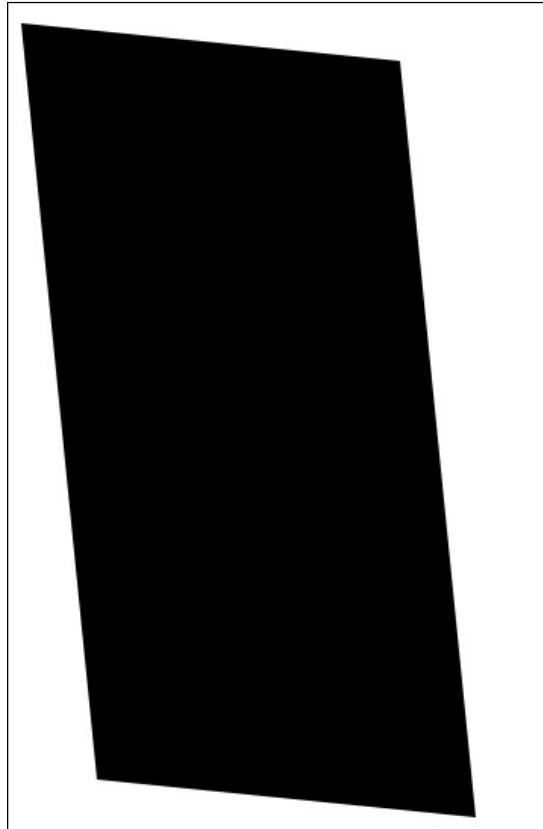
`vs` Verticaal schalen uitgedrukt in een percentage.

`hv` Horizontaal verplaatsen uitgedrukt in een x-coördinaat.

`vv` Verticaal verplaatsen uitgedrukt in een y-coördinaat.

Voorbeeld

```
ctx.transform(5,0.5,1,10,15,30); //vervormingsparameters  
//instellen  
ctx.fillRect(10,10,50,50); //rechthoek omschrijven en vervormd  
//tekenen
```



13.3_vervormen.js

An advertisement for Coca-Cola Enterprises. The left side features a large red arrow pointing right. Inside the arrow, the text "I WANT CHALLENGES THAT DEMAND INNOVATIVE SOLUTIONS" is written in white, bold, uppercase letters. Below this, the text "MY TIME IS NOW." is also in white, bold, uppercase letters. To the right of the arrow is a black and white portrait of a man looking upwards. At the bottom left, the Coca-Cola Enterprises logo is displayed in its signature script font, followed by the word "Thirst." in a bold, sans-serif font. The overall design is dynamic, using the red arrow to guide the eye towards the portrait and the brand message.

Download free eBooks at bookboon.com



Click on the ad to read more

Met de vervormtool kan je naar hartenlust experimenteren. Trek scheef, vergroot of verklein, alle instellingen hangen van je doel af. Zorg er wel voor dat je de percentages uitdrukt met een waarde tussen nul en één.

13.4 Verplaatsen

Syntax

Wanneer je het nulpunt, coördinaat `(0, 0)`, op de canvas wil verleggen, dan kan je `translate()` gebruiken. Die truc komt soms handig van pas om nauwkeurig te kunnen tekenen, zonder dat je al te veel rekenwerk hebt.

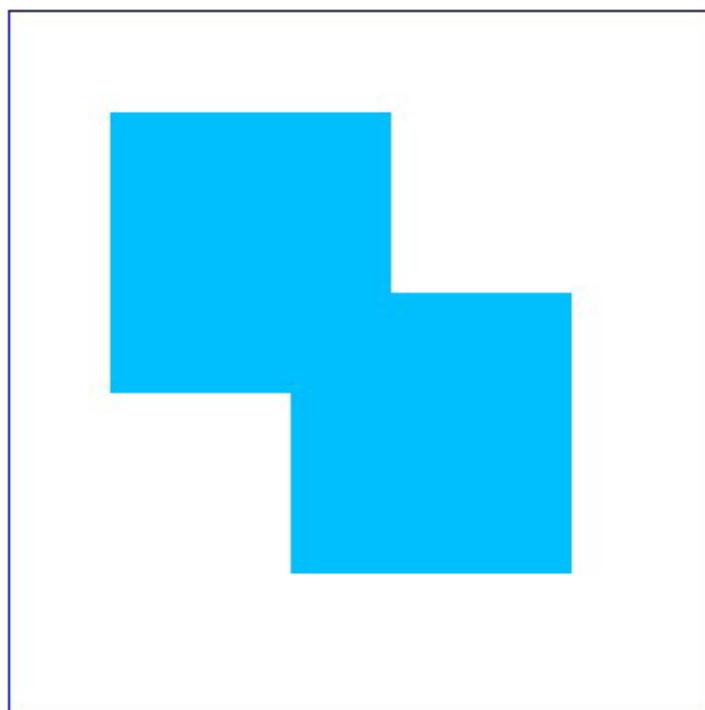
```
ctx.translate(x, y);
```

x De x-coördinaat van het nieuwe nulpunt.

y De y-coördinaat van het nieuwe nulpunt.

Voorbeeld

```
ctx.fillStyle = "deepskyblue"; //vulkleur  
ctx.fillRect(50, 50, 140, 140); //gevulde rechthoek tekenen  
ctx.translate(90, 90); //verplaatsen van het nulpunt  
ctx.fillRect(50, 50, 140, 140); //gevulde rechthoek tekenen
```



13.4_verplaatsen.js

De bovenstaande code levert twee identieke rechthoeken op. Alleen begint de eerste rechthoek linksboven in de canvas op coördinaat $(50, 50)$, terwijl de tweede rechthoek beginpunt $(140, 140)$ op de canvas krijgt, ook al staan de waarden voor die tweede rechthoek op $(50, 50)$. Die $(90, 90)$ is namelijk het nieuwe nulpunt. Daarvoor zorgt de onderstaande lijn.

```
ctx.translate(90, 90);
```

Vanaf dat moment is je canvas eigenlijk een stukje kleiner geworden.

13.5 Reset

Hou er rekening mee dat *alle* objecten die je na de `scale()`-methode, `rotate()`, `transform()` of `translate()` invoegt, getransformeerd worden. Wil je de oorspronkelijke waarden terug plaatsen? Pas dan de `setTransform()`-methode toe.

Syntax

De methode `transform()` ken je al. Daarmee kan je een vorm vervormen. Echter zie je dat er ook een methode `setTransform()` bestaat. Die methode reset alle parameters van de `transform()`-methode én roept de `transform()` opnieuw aan met de parameters die je aan `setTransform()` doorgeeft. Als je dat weet, dan kan je die methode ook als een resetter gebruiken.

```
ctx.setTransform(hs, hh, vh, vs, hv, vv);
```

Je ziet dat de argumenten identiek zijn aan die van `transform()`.

<code>hs</code>	Horizontaal schalen uitgedrukt in een percentage.
<code>hh</code>	Horizontale helling uitgedrukt in een percentage.
<code>vh</code>	Verticale helling uitgedrukt in een percentage.
<code>vs</code>	Verticaal schalen uitgedrukt in een percentage.
<code>hv</code>	Horizontaal verplaatsen uitgedrukt in een x-coördinaat.
<code>vv</code>	Verticaal verplaatsen uitgedrukt in een y-coördinaat.

Voorbeeld

Wil je bijvoorbeeld de scale() -methode resetten?

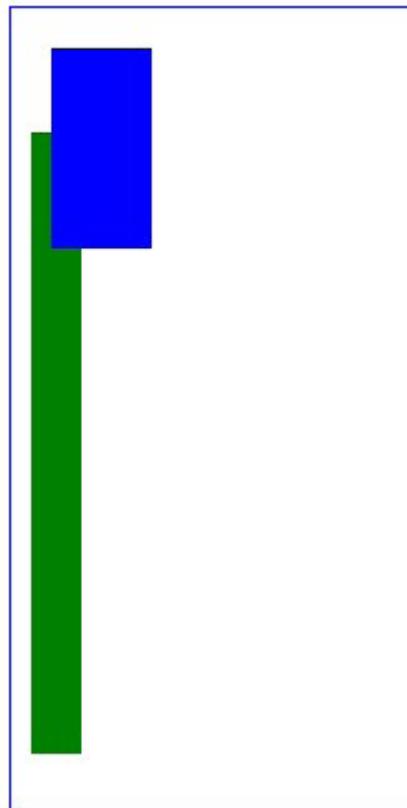
```
ctx.scale(0.5, 3.1); //schalen instellen  
ctx.fillStyle = "green"; //vulkleur instellen  
ctx.fillRect(20, 20, 50, 100); //ingekleurde rechthoek tekenen  
ctx.setTransform(1, 0, 0, 1, 0, 0); //reset van het schalen  
ctx.fillStyle = "blue"; //vulkleur instellen  
ctx.fillRect(20, 20, 50, 100); //ingekleurde rechthoek tekenen
```

Start jij jouw carrière als Junior Product Developer Innovatieve Printsystemen?

(Bijna) klaar met je WO-opleiding? Start bij Océ als Product Developer en werk aan inkten, printheads en print processen van de toekomst! Interesse? [Lees hier meer!](#)

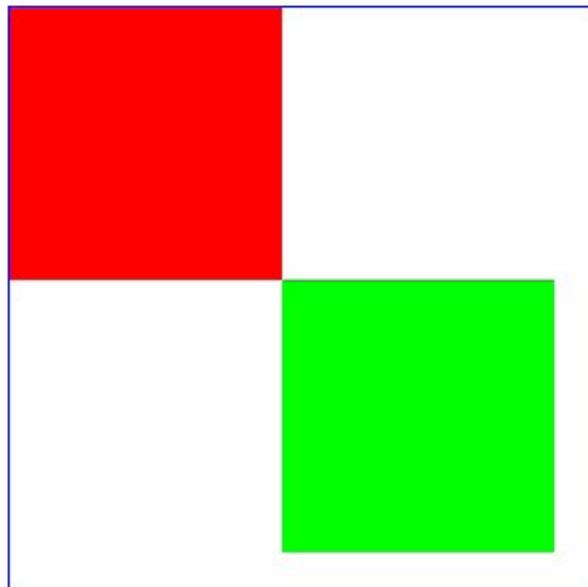
océ
A CANON COMPANY



**13.5_resetScale.js**

Wil je de `translate()` resetten en het nulpunt terug op zijn oorspronkelijke plek leggen? Dan haal je de `setTransform()`-methode boven.

```
ctx.translate(90,90); //nulpunt verplaatsen  
ctx.fillStyle = "#00FF00"; //groene vulkleur  
ctx.fillRect(50, 50, 140, 140); //ingekleurde rechthoek tekenen  
ctx.setTransform(1,0,0,1,0,0); //nulpunt terug op coördinaat 0,0  
leggen  
ctx.fillStyle = "#FF0000"; //rode vulkleur  
ctx.fillRect(0,0,140,140); //ingekleurde rechthoek tekenen
```



13.5_resetTranslate.js

Eerst verleg je het nulpunt. Dan teken je de groene rechthoek. Vervolgens reset je de instellingen , zodat de rode rechthoek dan terug helemaal linksboven in de canvas staat.



Career @ total.nl

In Nederland bestaat de Total Groep uit meer dan **10 bedrijven** die bij elkaar ruim **2.500 medewerkers** in dienst hebben. Zij houden zich bezig met **exploratie en productie van aardgas, raffinage van ruwe olie**, de verkoop van **olieproducten** en **chemie**. Total produceert op een **verantwoorde manier energie en producten** die nodig zijn in onze maatschappij, daarbijlettend op de **besparing** van energie en de **bescherming** van het milieu. Vanuit het kantoor in **Den Haag** verkopen wij brandstoffen en smeermiddelen. Dit in een **uitdagende markt** waarin wij als organisatie continu moeten **ontwikkelen** en **vernieuwen**.

Deze markt vraagt om **getalenteerde medewerkers!**

Check www.total.nl



Total kies je niet toevallig **TOTAL**

14 Afbeeldingen

Op de canvas kan je niet alleen tekenen, ook toon je in een handomdraai bestaande afbeeldingen, zoals jpg's of png's. Daarvoor beschikt de context over een methode `drawImage()`.

14.1 Laden en tonen

Syntax

```
ctx.drawImage(afbeelding, x, y);
```

`afbeelding` Het pad en de bestandsnaam van de afbeelding.

`x` De x-coördinaat van de linkerbovenhoek van de afbeelding.

`y` De y-coördinaat van de linkerbovenhoek van de afbeelding.

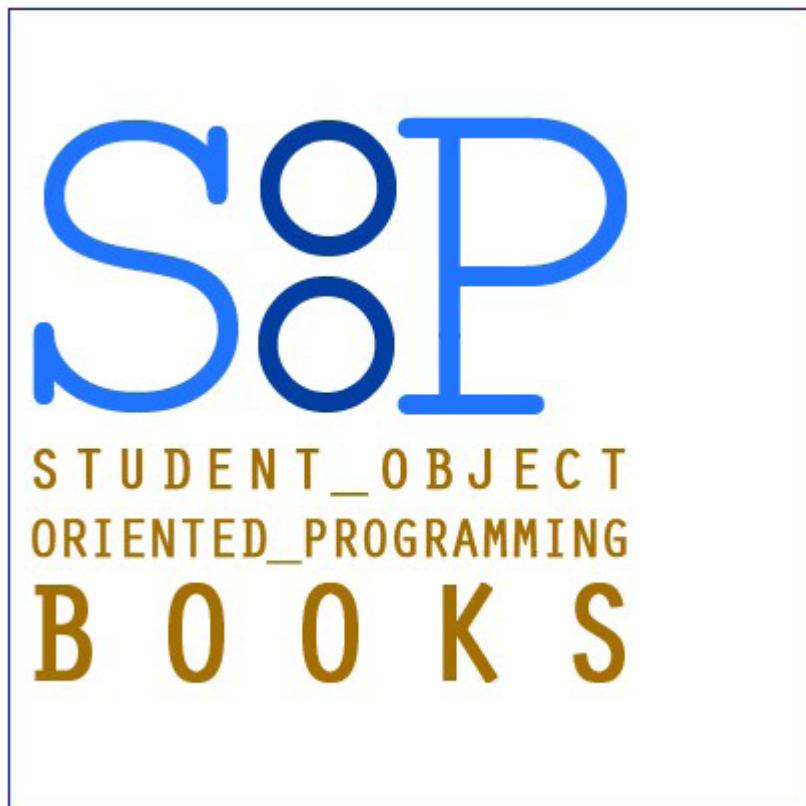
De methode `drawImage()` bezit drie argumenten. Ten eerste geef je de naam van de afbeelding die je wil tonen, op. Dat kan in theorie de bestandsnaam zijn, toch zal dat meestal een objectnaam inhouden.

Vervolgens bepaal je met de x- en de y-coördinaat de positie van de linkerbovenhoek van de afbeelding. Die coördinaat is relatief ten opzichte van de nulpositie van je canvas.

Voorbeeld

De `drawImage()`-methode verwacht als argument een object met daarin een afbeelding. Daarom moet je de te tonen afbeelding eerst laden. Pas wanneer die afbeelding volledig in het geheugen zit, kan `drawImage()` aan de slag.

```
var afbeelding = new Image(); //afbeeldingobject aanmaken
afbeelding.src = "SoopbooksLogo.png"; //bestand toekennen
afbeelding.onload = function() {
    //anonieme functie die pas uitgevoerd wordt nadat de afbeelding
    //volledig geladen is
    ctx.drawImage(afbeelding, 10, 50); //afbeelding weergeven
}
```



14.1_ladenAfbeelding.js

**STUDY ENGINEERING AND SCIENCE
AT AALBORG UNIVERSITY IN DENMARK**

AIM FOR A BRIGHT FUTURE - MOVE THE WORLD!

- Highly ranked programmes
- Problem based learning and team work
- Close collaboration with industrial partners

Study Engineering and Science in Denmark – build the future and move the world!

APPLY NOW - DEADLINE 1 APRIL 2015

MORE INFORMATION ON MOVETHEWORLD.DK


AALBORG UNIVERSITY
DENMARK







**NO TUITION
FEE FOR
EU/EEA-CITIZENS**

Download free eBooks at bookboon.com

Eerst maak je een nieuwe variabele aan.

```
var afbeelding = new Image();
```

Die variabele vormt een object, JavaScript richt die namelijk in om alle eigenschappen en methodes die bij een afbeelding horen, te bewaren. Dat doe je door met de `new`-operator de constructor `Image()` van het JavaScript-object `Image` aan te roepen.

Constructor-wat-zeg-je?

Doen termen zoals constructor of operator je de wenkbrauwen fronsen? Lees er dan gerust even mijn ebook JavaScript en JQuery op na.

Zodra je eigen afbeeldingenobject bestaat, kan je gebruik maken van de eigenschap `src`. Daar plaats je de naam van de afbeelding, eventueel samen met een (relatief) pad. Je kan daarbij gebruik maken van jpg, png, gif, svg en bmp. Vervolgens ben je klaar om je afbeelding door te geven aan de `drawImage()`-methode.

```
afbeelding.onload = function() {
    ctx.drawImage(afbeelding, 10, 50);
}
```

Je ziet dat we daarbij een anonieme functie in het leven roepen. Je kan `drawImage()` namelijk alleen aanroepen wanneer de afbeelding volledig in het geheugen geladen is. Daarom voert je `drawImage()` pas uit wanneer `onload` groen licht geeft. Die `onload` voert de anonieme functie pas uit wanneer de afbeelding `SoopbooksLogo.png` in je object `afbeelding` zit. Doe je die `onload` niet, dan zal de methode `drawImage()` de afbeelding `logo.png` niet kunnen vinden. De browser voert de lijnen code namelijk sneller uit, dan dat JavaScript de afbeelding in het geheugen kan laden. Daarom zou de browser sneller bij de lijn `ctx.drawImage(afbeelding, 10, 50)` aankomen, dan dat `SoopbooksLogo.png` in het geheugen zit.

14.2 Grootte

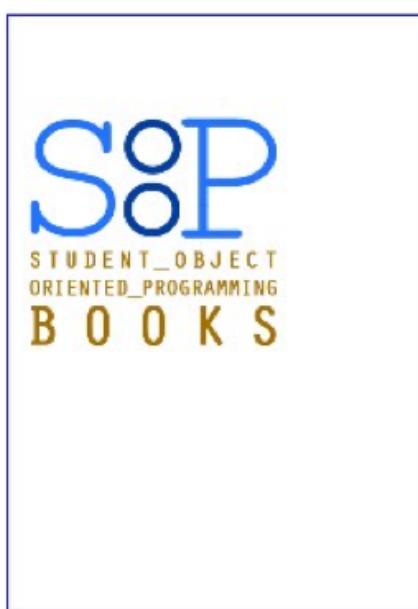
Syntax

Bij de bovenstaande argumenten kan je ook nog waarden toevoegen om de grootte van de afbeelding te bepalen – een overloaded methode. Op die manier kan je een afbeelding schalen.

```
ctx.drawImage(afbeelding, x, y, breedte, hoogte);
```

Voorbeeld

```
var afbeelding = new Image(); //nieuw afbeeldingobject
afbeelding.src = "logo.png"; //bestand toekennen
afbeelding.onload = function() { //anonieme functie: wachten op
    //laden
    ctx.drawImage(afbeelding,10,50,120,120); //afbeelding
    //tonen
}
```



14.2_schalenAfbeelding.js

In het bovenstaande voorbeeld krijgt de afbeelding de verhouding 120,120 mee.

14.3 Uitsnijden

Syntax

Moet je steeds je volledige afbeelding tonen? Nee, dat hoeft niet. Je kan afbeeldingen ook uitsnijden op de canvas. Zo kan je slechts een deel van de volledige afbeelding tonen. Daarvoor gebruik je opnieuw een overloaded methode. Die bezit een pak meer argumenten. Meer zelfs, de argumenten die de linkerbovenhoek van de afbeelding aanduiden, zijn van plaats veranderd.

```
ctx.drawImage(naamAfbeelding,cx,cy,cb,ch,x,y,b,h);
```

We overlopen de negen argumenten één voor één.

naamAfbeelding	Naam en pad van de afbeelding.
cx	De x-coördinaat van het clipping-punt.
cy	De y-coördinaat van het clipping-punt.
cb	De breedte van de afbeelding in de clippingzone.
ch	De hoogte van de afbeelding in de clippingzone.
x	De x-coördinaat van de linkerbovenhoek van de afbeelding.
y	De y-coördinaat van de linkerbovenhoek van de afbeelding.
b	Te gebruiken breedte van de afbeelding.
h	Te gebruiken hoogte van de afbeelding.

2014 Accenture. All rights reserved.

future > present

Be greater than.

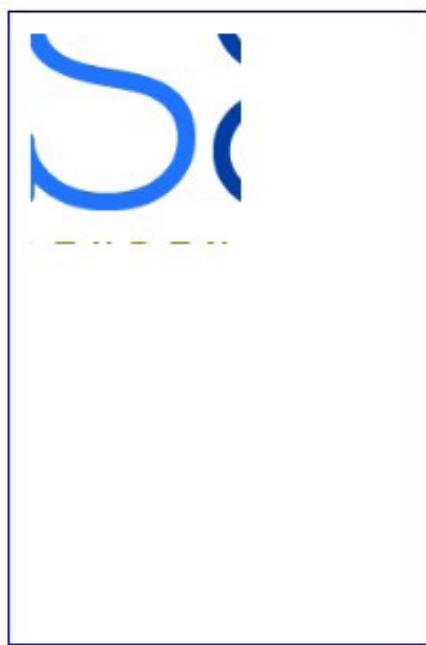
Dit is jouw kans om bij een organisatie te werken die je meer mogelijkheden, meer uitdaging en meer voldoening biedt.

Ga naar accenture.nl/carriere

accenture
High performance. Delivered.

Voorbeeld

```
var afbeelding = new Image(); //object voor de afbeelding  
                        //declareren  
afbeelding.src = "logo.png"; //bestand aan object toekennen  
afbeelding.onload = function() {  
//anonieme functie die wacht totdat afbeelding in het geheugen  
//zit  
    ctx.drawImage(afbeelding,10,50,120,120,10,10,100,100);  
}
```



14.3_uitsnijdenAfbeelding.js

Je merkt, met wat rekenwerk en wat proberen, snij je zo een stuk van een afbeelding uit. Dat kan je bijvoorbeeld gebruiken om de surfer op een afbeelding te laten inzoomen, wanneer je deze techniek met een schaalmethode combineert.

15 Effecten

Op de canvas kan je enkele effecten toepassen. Denk maar aan schaduw en transparantie.

15.1 Schaduw

Aan figuren kan je een slagschaduw toevoegen. Zo lijkt het alsof je diepte zicht creëert. Daarbij kan je alle kleuren toepassen en kan je de schaduw aan de randen afzwakken.

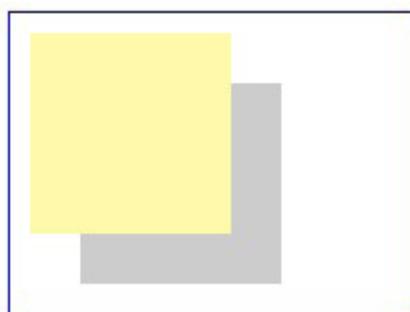
Syntax

```
ctx.shadowColor = "kleur";
ctx.shadowOffsetX = getal;
ctx.shadowOffsetY = getal;
```

Voorbeeld

Binnen de canvas kan je een schaduw aan een figuur toekennen. Daarvoor heb je een aantal eigenschappen van de context nodig. Een voorbeeld.

```
ctx.shadowColor = "#ccc"; //grijze kleur voor schaduw
ctx.shadowOffsetX = 25; //verschuiven schaduwvlak x-as
ctx.shadowOffsetY = 25; //verschuiven schaduwvlak y-as
ctx.fillStyle = "#FFFAAA"; //vulkleur
ctx.fillRect(10,10,100,100); //vierkant met vulkleur
```



15.1_schaduwRect.js

15.1.1 Lagen

Stel eerst de schaduwwaarden in en omschrijf dan pas de bovenliggende figuur. Doe je dat omgekeerd, dan zal je schaduw niet verschijnen. Respecteer de logische volgorde van het lagenmodel.

Wat kan je allemaal instellen? Bepaal eerst de kleur van het schaduwvlak.

```
ctx.shadowColor = "#ccc"; //grijze kleur voor schaduw
```

Die schaduw is eigenlijk een vlak dat onder de bijbehorende figuur gelegd wordt. Maar wat is de bijbehorende figuur? In het bovenstaand voorbeeld defnieer je eerst een schaduw en daarna volgt een rechthoek. Dan koppelt JavaScript die schaduw meteen aan die rechthoek.

Daarna geef je de coördinaat voor de linkerbovenhoek van het schaduwvlak op. Die coördinaat druk je niet relatief ten opzichte van het nulpunt van de canvas uit, wel relatief ten opzichte van de bijbehorende figuur.

```
ctx.shadowOffsetX = 25; //verschuiven schaduwvlak x-as  
ctx.shadowOffsetY = 25; //verschuiven schaduwvlak y-as
```

Wanneer je op de x-as – shadowOffsetX – een negatieve waarde ingeef, dan zal je schaduwvlak naar links verschuiven. Plaats je voor de y-as-waarde – shadowOffsetY – een minteken, dan kruipt de schaduw naar boven.

In the past four years we have drilled

89,000 km

That's more than **twice** around the world.

Who are we?
 We are the world's largest oilfield services company¹. Working globally—often in remote and challenging locations—we invent, design, engineer, and apply technology to help our customers find and produce oil and gas safely.

Who are we looking for?
 Every year, we need thousands of graduates to begin dynamic careers in the following domains:

- Engineering, Research and Operations
- Geoscience and Petrotechnical
- Commercial and Business

What will you be?

Schlumberger

¹Based on Fortune 500 ranking 2011. Copyright © 2015 Schlumberger. All rights reserved.

15.1.2 Vervagen

Een gewone schaduw ziet er behoorlijk hard uit. Toch kan je ook een blur inschakelen.

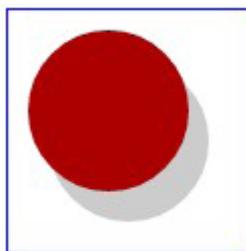
```
ctx.shadowBlur = 5; //vervagen van de kleur
```

Zo'n blur zorgt ervoor dat de rand van het schaduwvlak wat minder scherp gemaakt wordt.

15.1.3 Andere figuren

Ook kan je een schaduw aan andere figuren toevoegen, zoals een cirkel.

```
ctx.shadowOffsetX = 10; //verschuiving op x-as
ctx.shadowOffsetY = 15; //verschuiving op y-as
ctx.shadowColor = "#CCC"; //vulkleur voor schaduw
ctx.arc(50, 50, 40, 0, 2 * Math.PI, false); //cirkel
ctx.fillStyle = "#AB0000"; //vulkleur cirkel
ctx.fill(); //cirkel weergeven
```



15.1.3_schaduwCirkel.js

15.2 Transparantie

Daarnaast kan je figuren ook transparant maken. Daarbij kan je kiezen van lichttransparant tot bijna volledig doorzichtig.

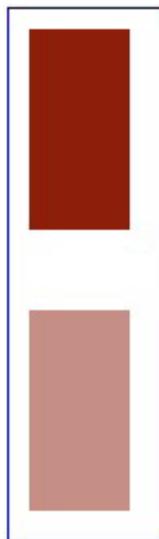
Syntax

```
ctx.globalAlpha = decimaalGetal;
```

decimaalGetal Een waarde tussen 0 en 1.

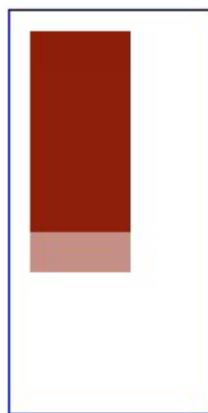
Voorbeeld

```
ctx.fillStyle = "rgb(140,30,10)"; //vulkleur instellen
ctx.fillRect(10,10,50,100); //ingekleurde rechthoek tekenen
ctx.globalAlpha = 0.5; //transparantiewaarde
ctx.fillStyle = "rgb(140,30,10)"; //vulkleur instellen
ctx.fillRect(10,150,50,100); //ingekleurde rechthoek tekenen
```

**15.2_transparantieLos.js**

In het bovenstaande voorbeeld zie je twee identieke rechthoeken met dezelfde kleur. Echter is de tweede rechthoek half doorschijnend gemaakt. Bij het onderstaande voorbeeld overlappen beide rechthoeken elkaar. Zo zie je meteen het laageffect.

```
ctx.fill = "rgb(140,30,10)";  
ctx.fillRect(10,10,50,100);  
ctx.globalAlpha = 0.5;  
ctx.fill = "rgb(140,30,10)";  
ctx.fillRect(10,30,50,100);
```

**15.2_transparantieRakend.js**

De ene rechthoek ligt bovenop de andere rechthoek.

16 Opslaan als afbeeldingbestand

Wanneer je uitermate trots bent op al je tekenwerk, dan kan je al die codeerlijnen zonder veel moeite als een jpg- of een png-afbeelding opslaan. Daarbij kan je twee wegen volgen. Ofwel regel je dat in code, ofwel vraag je de browser om hulp.

16.1 Code

Om een illustratie uit de canvas naar een afbeeldingsbestand te bewaren, spreek je de canvas rechtstreeks aan.

```
ctx.fillRect(10,10,100,100); //figuren tekenen...
window.location.href = c.toDataURL("image/png")
    .replace("image/png", "image/octet-
stream");
//stel een png-bestand samen en start de download ervan in de
//browser
```

De tweede, derde en vierde lijn horen op één enkele lijn te staan. Plaats er géén enters tussen.

The advertisement features a young man with curly hair smiling and holding a coffee cup, set against a blurred background of an office or study area. The text is overlaid on the right side of the image.

AARHUS BSS SCHOOL OF BUSINESS AND SOCIAL SCIENCES AARHUS UNIVERSITY

EQUIS ACCREDITED

AMBA ACCREDITED

EFMD EQUIS ACCREDITED

**Bachelor of Engineering -
Global Management
and Manufacturing**

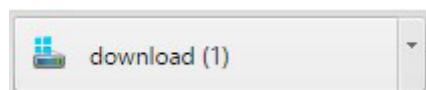
Aarhus BSS is part of Aarhus University in Denmark. It is ranked among the top 100 universities in the world due to its high standards in both education and research.

We offer English-taught programmes at all educational levels: Bachelor's, Master's, continuing education (MBA) and PhD programmes.

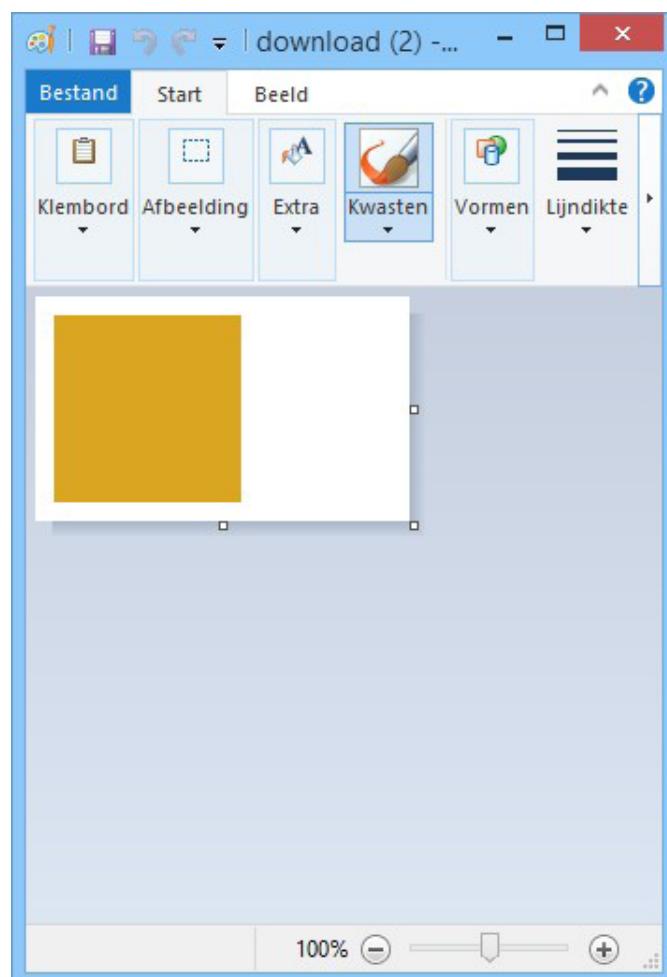
Read more
bss.au.dk/international

**16.1_naarAfbeelding.js**

Wanneer je de bovenstaande code in een browser uitvoert, start die automatisch een download op. Hieronder zie je het resultaat in Google Chrome.

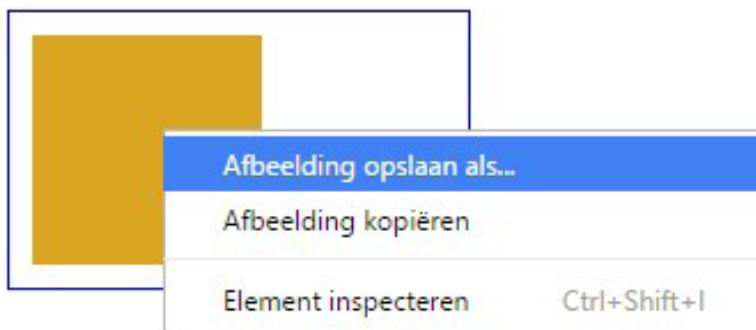


Dat bestand kan je vervolgens in een tekenpakket zoals Paint openen en bewerken.



16.2 Browser

Echter hoeft je niet te programmeren om je tekenwerk te bewaren. Surf met de browser naar je html-pagina. Wanneer je dan het resultaat van de canvas in de browser bekijkt, kan je in vele browsers daarop rechts klikken en de illustratie als een jpg of png bewaren. Het screenshot hieronder is in Google Chrome genomen.





Verbind jezelf aan een veelzijdige technische carrière.

De realisatie van de A2 landtunnel. Het volledige energiesysteem van het Oosterdokseiland. De projecten van Cofely zijn al snel bijzonder. Als grootste technisch dienstverlener van Nederland kun je bij ons innovatieve oplossingen creëren voor bijvoorbeeld Shell, ASML, Heineken en Rijkswaterstaat.

Of je nu technisch specialist bent en je vakinhoudelijk wilt verdiepen, of jezelf in een leidinggevende rol wilt ontwikkelen: binnen onze veelzijdige organisatie kies je zelf welke kant je op wilt.

Cofely is groot maar persoonlijk, internationaal en toch dichtbij. Heb jij een passie voor techniek en de drive om het beste uit jezelf te halen? Ontdek dan hoe ver jij kunt komen op werkenbijcofely.nl.

Verbind jezelf aan **COFELY**
GDF SUEZ

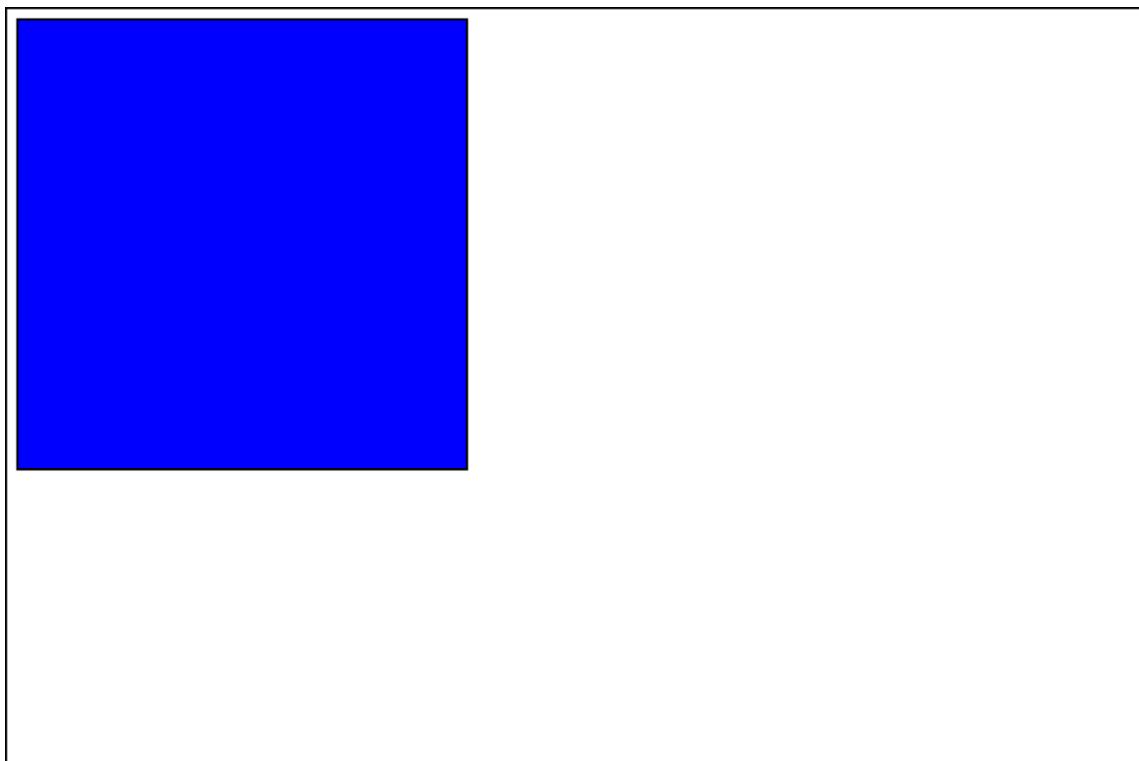
17 Animatie

Ziezo, tekenen kan je nu al. Je ontwerpt ondertussen knappe tekeningen in code, alleen de beweging ontbreekt nog. Die beweging voeg je toe met JavaScript. Daarbij kan je twee wegen bewandelen, de gevorderde weg met `requestAnimationFrame()` of het eenvoudigere pad van `setInterval()`.

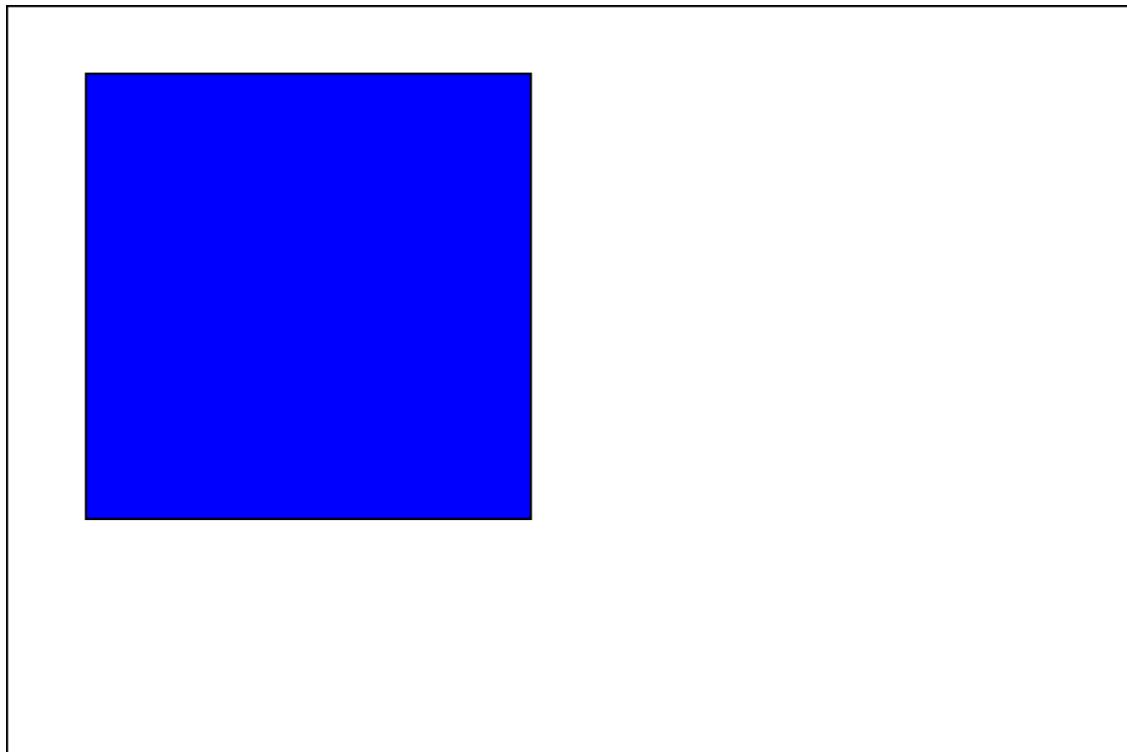
17.1 Frame

Je kent het principe van film. In plaats van bewegende beelden, plaats je heel veel foto's van dezelfde scène snel achter elkaar. Zo krijg je de illusie van een bewegend beeld. Bij animaties pas je dezelfde truc toe. Alleen werk je bij animaties niet met foto's, wel met zogeheten frames.

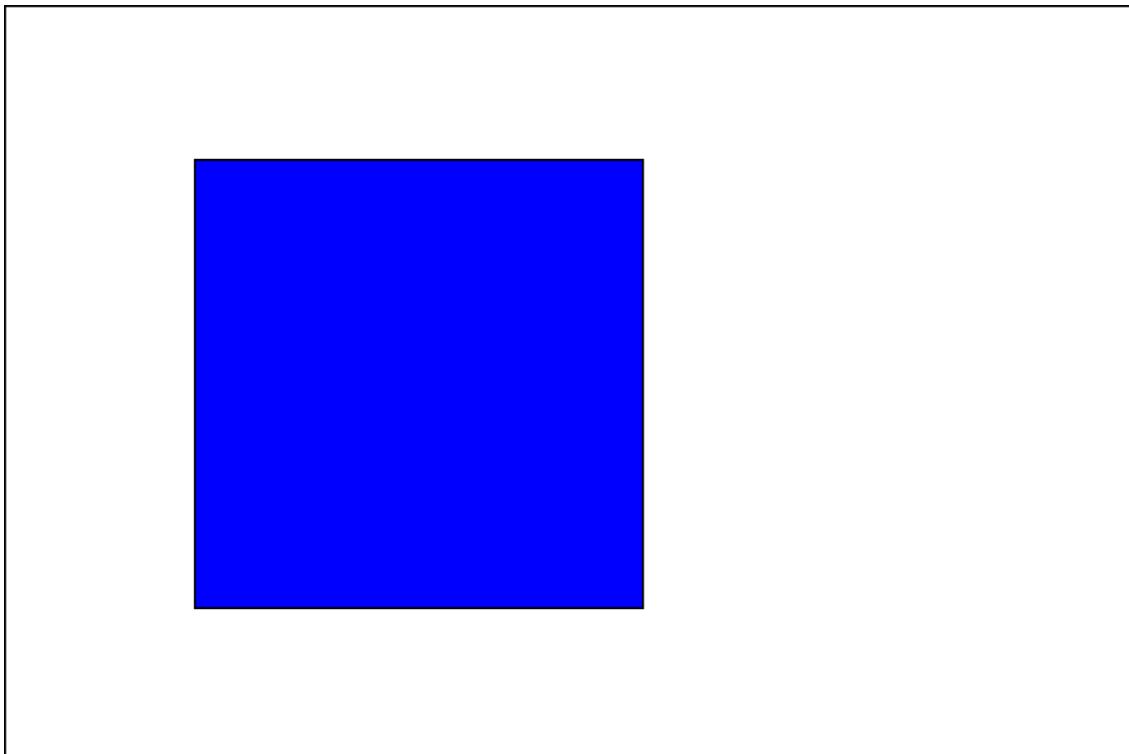
Een frame is een stilstaande illustratie. Bijvoorbeeld een vierkant. Dat vierkant staat bijvoorbeeld in de linkerbovenhoek van de canvas.



Neem vervolgens een tweede frame. Dat is opnieuw een stilstaande illustratie waarbij dat vierkant nu niet langer op coördinaat 0,0 staat, maar wel op bijvoorbeeld 15,15.



Vervolgens doe je nog eens hetzelfde en zet je het vierkant in het derde frame op 40,40.



Toon die drie afbeeldingen nu snel na elkaar, dan zal je het vierkant – weliswaar schokkerig – zien bewegen.

17.2 JavaScript

Met de klassieke animaties verzamel je dus een reeks van afbeeldingen. Met de canvas hoef je echter geen aparte afbeeldingen op te maken. Je kan figuren namelijk met code opbouwen. Vervolgens kan je de coördinaten van je figuur stapsgewijs wijzigen. Daardoor beweegt je figuur. Of tenminste, de surfer denkt dat de figuur beweegt. Maar hoe doe je dat?

```
ctx.rect(0,0,50,50);
```

Coördinaten van een figuur kan je vast inzetten – hardgecodeerd zoals hierboven – echter kan je die ook variabel maken.

```
var xPunt, yPunt, breedte, hoogte;
xPunt = 0;
yPunt = 0;
breedte = 50;
hoogte = 50;
ctx.rect(xPunt,yPunt,breedte,hoogte);
```



ANTEA GROUP

Van stad tot land, van water tot lucht; de ingenieurs en adviseurs van Antea Group dragen sinds jaar en dag bij aan onze leefomgeving. We ontwerpen bruggen en wegen, realiseren woonwijken en waterwerken. Maar we zijn ook betrokken bij thema's zoals milieu, veiligheid, assetmanagement en energie.

Wie zoeken wij? Het werk van Antea Group is enorm veelzijdig. We tekenen en rekenen, onderzoeken en adviseren, organiseren en realiseren. Dit doen we binnen verschillende werkvelden. Of je nu starter bent of senior, specialist of allrounder: in onze organisatie zijn er altijd mogelijkheden. We zoeken naar hbo'ers en wo'ers met oog voor de wereld om zich heen en passie voor hun werk.

WWW.WERKENBIJANTEAGROUP.NL

Knappe mensen. Mooi bedrijf.





Click on the ad to read more

Wanneer de argumenten van methodes in variabelen gegoten worden, kan je die aan de hand van andere JavaScript-instructies tijdens de uitvoer van de JavaScript-code wijzigen – de zogeheten runtime. Dan volstaat het om na het wijzigen van die parameters de inhoud van het scherm te hertekenen. Zo'n hertkening kan met de methode `requestAnimationFrame()` of met `setInterval()`. Wij spitten ze één voor één uit.

17.3 `requestAnimationFrame()`

Syntax

```
window.requestAnimationFrame(functienaam);
```

functienaam De naam van de functie die de acties voor de animaties bevat, zonder de ronde haakjes achter die functienaam.

Voorbeeld

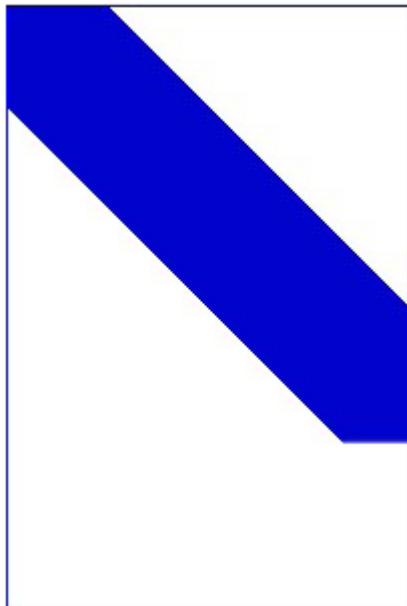
```
function bewegen() { //functie met acties voor animatie

    ctx.rect(xPunt,yPunt,breedte,hoogte); //omschrijf
                                         //rechthoek

    ctx.stroke(); //teken de rechthoek
    xPunt = xPunt + 1; //verhoog x-coördinaat met 1
    yPunt = yPunt + 1; //verhoog y-coördinaat met 1
    window.requestAnimationFrame(bewegen);
    //voer deze functie opnieuw uit en herschrijf het scherm

}

window.requestAnimationFrame(bewegen);
//voer deze functie uit en herschrijf het scherm
```

**17.3_requestAnimationFrame.js**

Dat ziet er al meteen knap gevorderd uit. Je roept namelijk binnen de functie bewegen() diezelfde functie bewegen() terug op... Dat heet een recursieve functie. De methode `requestAnimationFrame()` zorgt ervoor dat de functie bewegen() uitgevoerd wordt. In die functie updaten we de coördinaten van de rechthoek. De methode `requestAnimationFrame()` zorgt er vervolgens voor dat het canvas-scherf herschreven wordt – er wordt een nieuw frame getoond.

17.4 setInterval()

Voor de komst van de methode `requestAnimationFrame()` voerde je een animatie uit op basis van het ondertussen verouderde `setInterval()`. Omdat die manier nog veel toegepast wordt, tonen we ze hier ook.

Syntax

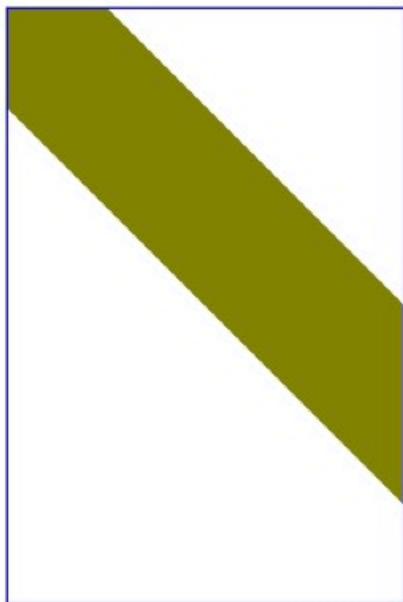
```
setInterval(functienaam, milliseconden)
```

functienaam De naam van de functie die de acties voor de animatie bevat. Let erop dat je geen ronde haakjes na de naam plaatst.

milliseconden Geef het aantal milliseconden met een geheel getal op waarna het canvas-scherf ververst moet worden.

Daarmee kan je op een soortgelijke manier als de `requestAnimationFrame()` de canvas om de zoveel milliseconden refreshen, echter is die methode arbeidsintensiever voor de processor.

```
function bewegen() {  
  
    ctx.rect(xPunt,yPunt,breedte,hoogte); //omschrijf  
                                         //rechthoek  
    ctx.stroke(); //teken de rechthoek  
    xPunt = xPunt + 1; //verhoog x-coördinaat met 1  
    yPunt = yPunt + 1; //verhoog y-coördinaat met 1  
    window.setInterval(bewegen);  
    //voer deze functie opnieuw uit en herschrijf het scherm  
  
}  
  
window.setInterval(bewegen,200);  
//voer deze functie uit en herschrijf het scherm
```



17.4_setInterval.js

Met `window.setInterval()` zeg je welke functie volgens een regelmatig interval uitgevoerd moet worden. Dat interval druk je in milliseconden uit. Voor het overige werkt deze methode soortgelijk als de `requestAnimationFrame()`.

18 Oefenen

Om het tekenen met code stevig onder de knie te krijgen, is het belangrijk dat je veel oefent en experimenteert. Start met eenvoudige tekeningen zoals een huisje of een autootje.

Onthoud daarbij dat dé oplossing niet bestaat. Je kan een vierkant bijvoorbeeld zowel als vier lijnen tekenen, als met de rechthoekmethode.

Op claimjekennis.soopbooks.com zie je verschillende opgaven staan. Echter kan je met je code vastlopen. Gebruik de element inspector van je browser – kijk bijvoorbeeld de console na op fouten – en spoor zo problemen op. Gebruik bovendien ook de volgende checklist.

18.1 Checklist

- Html-containers: heb je alle html-containers correct en op de juiste plek afgesloten?
- Puntkomma: sluit elke JavaScript-instructie met een puntkomma af.
- Aanhalingstekens: gebruik je de juiste rechte aanhalingstekens in je code? Sluit je ze ook correct af?
- Declaratie: declareer elke globale variabele steeds in de <head>.

WHILE YOU WERE SLEEPING...

INDIAN NUCLEAR POWER CORP
RUSSIAN ENTREPRENEURS
LUXURY GOODS MARKET LONDON
CHINA'S PREMIUM PLAYS
U.S. TRADE DEFICITS

www.fuqua.duke.edu/whileyouweresleeping

DUKE
THE FUQUA
SCHOOL
OF BUSINESS

Deel 3 – Svg

1 Doelen

Als je dit derde deel doorgenomen hebt, bereik je de onderstaande doelstellingen. Je kan op dat moment je kennis en vaardigheden ook even testen met de oefeningen aan het einde van dit deel.

Overigens vormen die oefeningen een absolute aanrader. Want svg en xml is oefenen, oefenen en oefenen. Alleen zo word je de heerster van vectoriële illustratie!



Illustratie 4 – Check aan het einde van dit deel of je een antwoord kan geven bij deze doelstellingen. soopbooks.com

1. Je kent de voor- en nadelen van svg.
2. Je weet dat svg een vectoriële afbeelding genereert.
3. Je kan svg-bestanden met een tekenpakket aanmaken.
4. Je begrijpt de dom-structuur van een svg-bestand.
5. Je kan een svg-container in de body aanmaken.
6. Je kan vormen tekenen, zoals een lijn, een rechthoek, een cirkel, een ellips, een veelhoek en een veelvoudige lijn.
7. Je kan het begrip path toelichten.
8. Je kan diverse soorten paden aanmaken en afsluiten.
9. Je kan teksten in een svg-container toevoegen.
10. Je kan teksten positioneren.
11. Je kan een tekst omtoveren tot een hyperlink.
12. Je kan een eenvoudige animatie met svg maken.

2 Inleiding

Binnen de canvas werk je met rasterafbeeldingen. Zulke afbeeldingen zijn een verzameling van losse pixels. Zo bestaat een lijn dan uit honderden pixels die aangrenzend zijn, dat weet je ondertussen al. Svg, ofwel *scalable vector graphics*, bouwt daarentegen alle figuren met wiskundige formules op. Zo teken je een rechte lijn daar op basis van een begin- en een eindpunt.

2.1 Vectoren

Een vectorillustratie is een afbeelding waarvan alle punten een welbepaalde samenhang kennen. Je werkt er met mathematische formules die figuren zoals lijnen, cirkels, veelhoeken of krommen beschrijven. Doordat je wiskunde gebruikt, ben je niet aan afmetingen gebonden. Wil je een figuur vergroten of verkleinen, dan rekent de svg-software elke keer de juiste pixelverhoudingen voor je uit. Daardoor krijg je steeds een haarscherpe figuur. Probeer maar eens in- of uit te zoomen op een svg. Je merkt geen enkel kwaliteitsverlies.



Winning awards is in our nature

Corbion Purac is proud to be a Top Employer in 2014 for the 5th consecutive year.

Corbion is the global market leader in lactic acid, lactic acid derivatives and lactides, and a leading company in functional blends containing enzymes, emulsifiers, minerals and vitamins. The company delivers high performance biobased products made from renewable resources and applied in global markets such as bakery, meat, pharmaceuticals and medical devices, home and personal care, packaging, automotive, coatings and resins. Its products have a differentiating functionality in all kinds of consumer products worldwide.

Would you like to know more about Corbion Purac or do you want to know more about the vacancies within Corbion Purac? Visit: www.jobsatcorbion.com



Join our biobased innovation engine



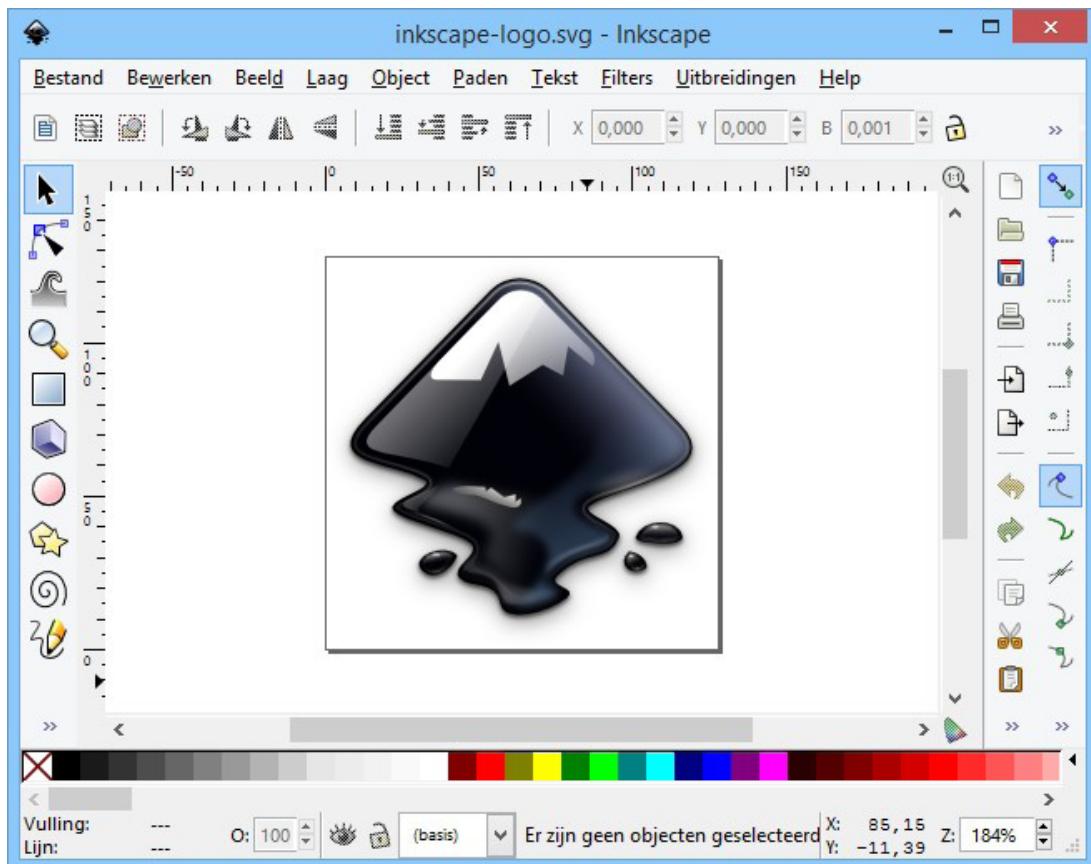
Download free eBooks at bookboon.com



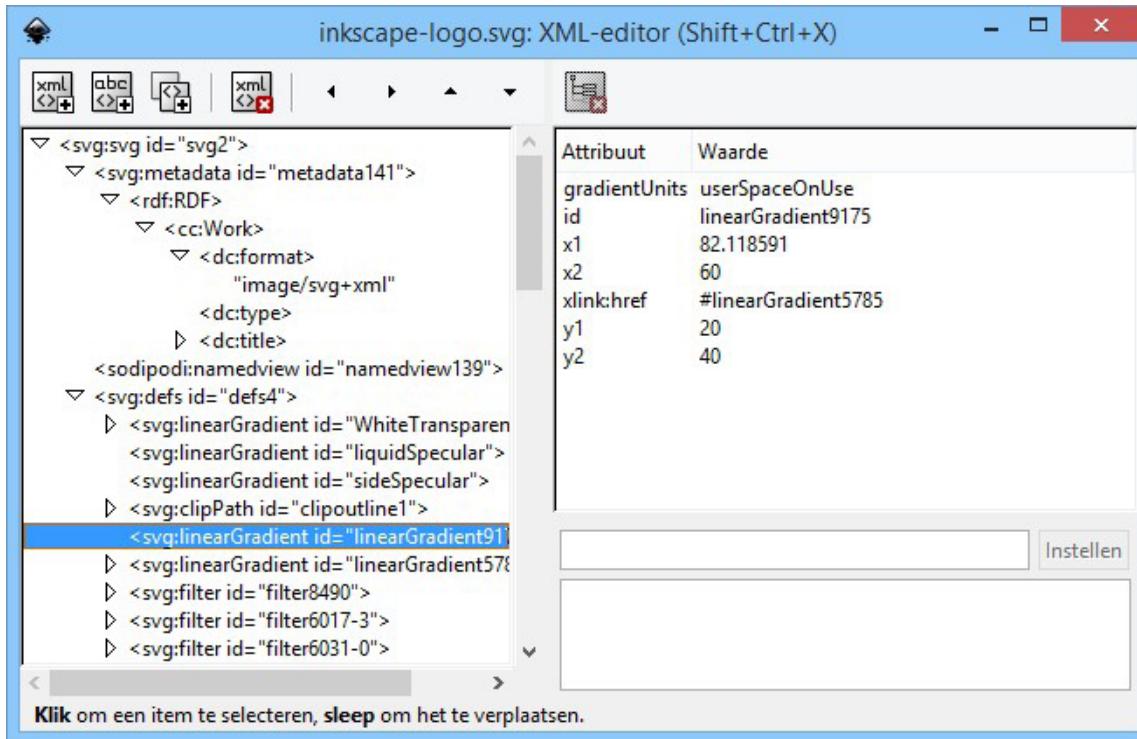
Click on the ad to read more

2.2 Inkscape

Svg-codes zijn fun, maar moet je een svg-bestand steeds met code opbouwen? Nee hoor, je kan dat ook op een zuiver grafische manier. Je zal namelijk ontdekken dat svg-codes behoorlijk ingewikkeld kunnen worden. Dan is een programma zoals Inkscape de meest efficiënte weg. Download het gratis programma Inkscape op <https://inkscape.org/nl/download/>. Dat programma kan een grafische gegenereerde illustratie prima in code vertalen. Daarom is het interessant om ook de code-achtergrond te kennen. Daarom behandelen we in dit deel de belangrijkste aspecten van svg.



Inkscape-logo in vectorieel tekenpakket Inkscape



Hetzelfde logo in svg-codes.

2.3 SVG-dom

Een svg-bestand kan je met xml-codes opbouwen. Die codes steunen op een dom, ofwel het document object model. Daarin vind je een overzicht van alle objecten.

Daardoor kan je alle eigenschappen en methodes gebruiken die aan een bepaald object gekoppeld zijn, ook al vermeld je die niet uitdrukkelijk in je html-code.

Objecten

Wil je meer over objecten en JavaScript weten? Kijk dan in mijn boek Javascript en JQuery.

3 <svg>-container

Binnen html kan je een svg-figuur invoeren door gebruik te maken van de <svg>-container. Voeg daarom een <svg>-container binnen de <body> in.

Syntax

```
<svg width="breedte" height="hoogte"></svg>
```

Met de <svg>-container baken je de zone binnen de <body> af, waarbinnen de vectoriële figuren geplaatst worden. Net zoals bij de <canvas> bepaal je de hoogte en de breedte met de argumenten width en height. We drukken graag nog eens op de schrijfwijze van die width – met dth – en height – met ght. Vergeet de container van <svg> ook niet af te sluiten.

width	De breedte van het werkveld.
height	De hoogte van het werkveld.



Julian Lienich, engineer

I can shape the future. Every day.

The E.ON graduate program requires my energy and creative input. In exchange I get to work with up-to-date technologies in a team that supports my professional development. What about you?

Your energy shapes the future.

www.eon-career.com

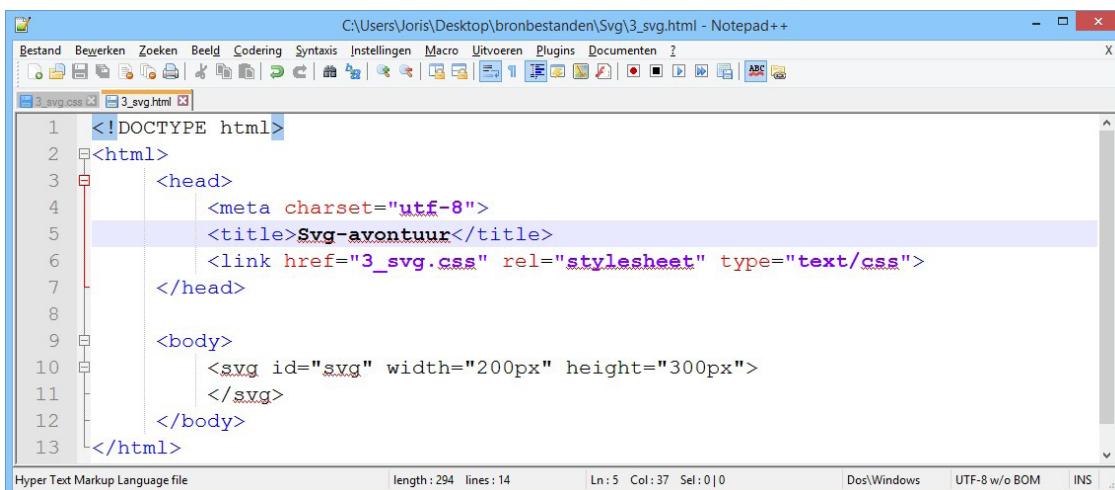
e-on



Click on the ad to read more

Voorbeeld

```
<svg width="600" height="400"></svg>
```



The screenshot shows a Notepad++ window with the file '3_svgContainer.html' open. The code is as follows:

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Svg-avontuur</title>
6     <link href="3_svg.css" rel="stylesheet" type="text/css">
7   </head>
8
9   <body>
10    <svg id="svg" width="200px" height="300px">
11    </svg>
12  </body>
13 </html>

```

The Notepad++ interface includes a toolbar at the top, a status bar at the bottom, and various tabs and icons.

3_svgContainer.html in Notepad++

In de bovenstaande code maak je een rechthoekige zone waarbinnen de svg-instructies uitgevoerd kunnen worden. Die zone krijgt dankzij css een blauwe omkadering mee.

Merk je in de kleurcodering van Notepad++ op dat de <svg>-container niet herkend wordt als html-element? Maak je daar echter geen zorgen over. Zodra je de code in een browser runt, werkt je svg perfect.

Tussen de container kan je ook een tekst plaatsen voor oudere browsers die svg niet herkennen.

```
<svg width="breedte" height="hoogte">Helaas, je browser
ondersteunt geen svg.</svg>
```

Probeert een surfer je svg te bekijken met een browser die nog nooit van html5 gehoord heeft, dan negeert die browser alle svg-specifieke tags en toont die enkel de foutbericht.

4 Vormen

Op basis van wiskundige formules kan je diverse figuren tekenen. Echter hoeft je die formules niet te kennen, daar zorgt SVG wel zelf voor. Het volstaat om code te schrijven.

4.1 Lijn

Syntax

De lijn is een eenvoudige figuur. Je verbindt een beginpunt met een eindpunt.

```
<line x1="getal" y1="getal" x2="getal" y2="getal"  
      style="stijlopties">
```

x1 De x-coördinaat van het beginpunt van de eerste lijn.

y1 De y-coördinaat van het eindpunt van de eerste lijn.

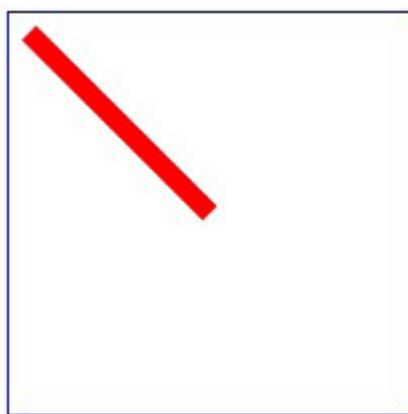
x2 De x-coördinaat van het beginpunt van de tweede lijn.

y2 De y-coördinaat van het eindpunt van de tweede lijn.

style De stijlkenmerken in CSS.

Voorbeeld

```
<line x1="10" y1="10" x2="100" y2="100" style="stroke:red;  
stroke-width:10px;">
```



4.1_lijn.html

De bovenstaande code levert een rode lijn op tussen coördinaat 10,10 en 100,100. Met de stijleigenschappen `stroke` en `stroke-width` bepaal je de kleur rood en de lijndikte.

4.2 Rechthoek

Syntax

Een rechthoek heeft een beginpunt nodig – de hoek linksboven – en een hoogte en een breedte.

```
<rect x="getal" y="getal" width="breedte" height="hoogte"
      style="stijlopties">
```

`x` De x-coördinaat van de linkerbovenhoek.

`y` De y-coördinaat van de linkerbovenhoek.

`width` Breedte van de rechthoek.

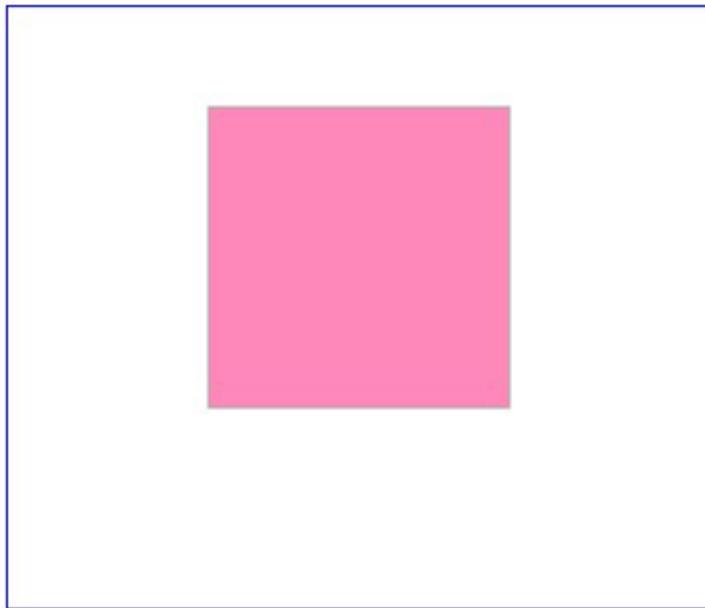
`height` Hoogte van de rechthoek.

`style` Stijlkenmerken in css.



Voorbeeld

```
<rect x="100" y="50" width="150" height="150"  
      style="fill:#F8B;stroke:#AAA;">
```



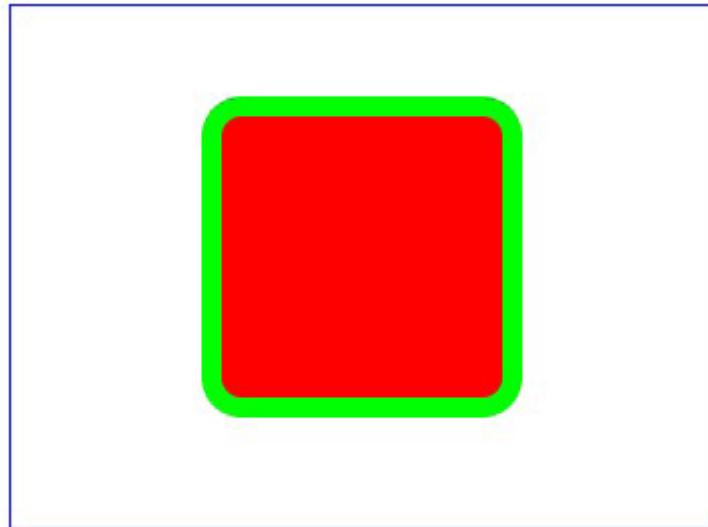
4.2_rechthoek.html

In de bovenstaande lijn ontdek je hoe er een vierkant geschetst wordt. Dat vierkant begint met de linkerbovenhoek op coördinaat 100,50 en bezit een vulkleur en een lijnkleur. Je kan die stijleigenschappen verder nog uitbreiden met alle css-eigenschappen die op een rechthoek toegepast kunnen worden.

Voeg je de argumenten `rx` en `ry` toe, dan zorg je voor een afronding van de hoeken.

```
<rect x="100" y="50" width="150" height="150" rx="15" ry="15"  
      style="fill:#F00;stroke:#0F0;stroke-width:10px">
```

Met twee stralen, `rx` en `ry`, kan je de hoeken van de rechthoek afronden. Daarvoor past svg een ellips toe. Daarom heb je een straal voor de x-as nodig (`rx`) en een straal voor de y-as (`ry`). De omlijning kreeg breedte 10 pixels mee, de vulkleur staat op rood, terwijl de rand groen is.



4.2_rechthoekAfgerond.html

4.3 Cirkel

Syntax

```
<circle cx="getal" cy="getal" r="getal" style="stijlopties">
```

cx De x-coördinaat van het middelpunt van de cirkel.

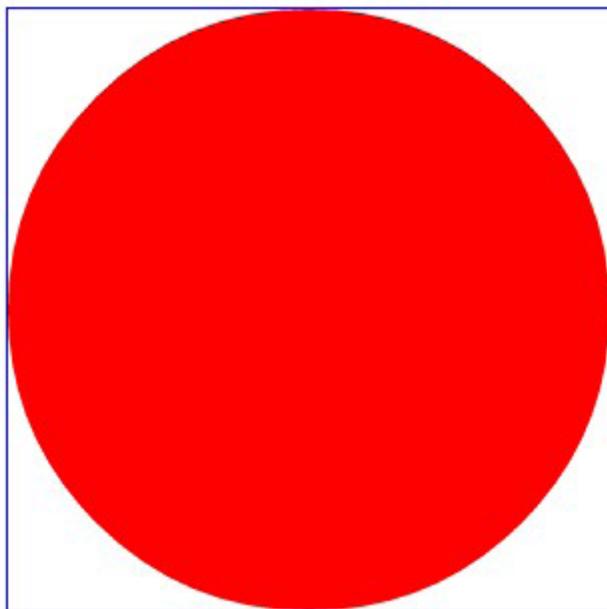
cy De y-coördinaat van het middelpunt van de cirkel.

r De straal van de cirkel.

style De stijlkenmerken in css.

Voorbeeld

```
<circle cx="150" cy="150" r="15" style="fill:red">
```

**4.3_cirkel.html**

I joined MITAS because
I wanted **real responsibility**

The Graduate Programme
for Engineers and Geoscientists
www.discovermitas.com

Month 16

I was a construction supervisor in the North Sea advising and helping foremen solve problems

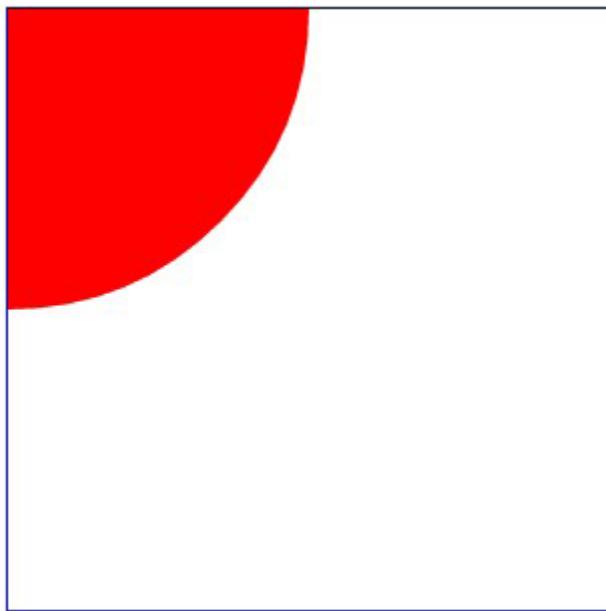
Real work
International opportunities
Three work placements

MAERSK

De cirkel heeft als middelpunt coördinaat 150,150 en een straal van 15 pixels. De cirkel krijgt een rode vulkleur.

Tussen haakjes, je zou verwachten dat je `cx` (circle x-as) door `c` en `cy` (circle y-as) door `y` zou kunnen vervangen. Maar dat loopt schijnbaar fout. Toch handelt de browser heel rechtdoorzee. Een voorbeeld.

```
<circle x="150" y="150" r="15" style="fill:red">
```



4.3_cirkelFout.html

Plots staat de cirkel in de bovenhoek. Toch is dat logisch. Svg zoekt voor het middelpunt van de cirkel naar de attributen `cx` en `cy`. De argumenten `x` en `y` kent svg bij `circle` niet, daardoor negeert svg die argumenten botweg. Wanneer de argumenten `cx` en `cy` vervolgens ontbreken, zet svg het middelpunt standaard op coördinaat 0,0. En dat zie je ook in de bovenstaande afbeelding, de cirkel staat met het middelpunt netjes op dat coördinaat geparkeerd.

4.4 Ellips

Syntax

Een ellips omschrijf je door een aparte straal voor de x- en de y-as te voorzien.

```
<ellipse cx="getal" cy="getal" rx="getal" ry="getal"
          style="stijlopties">
```

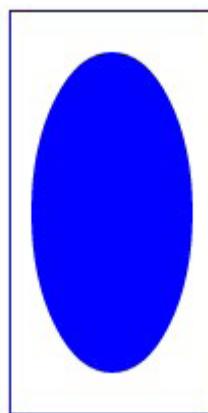
`cx` De x-coördinaat van het middelpunt van de cirkel.

`cy` De y-coördinaat van het middelpunt van de cirkel.

rx	De horizontale straal.
ry	De verticale straal.
style	De stijlkenmerken in css.

Voorbeeld

```
<ellipse cx="50" cy="100" rx="40" ry="80" style="fill:blue;">
```



4.4_ellips.html

In dit voorbeeld maak je een blauwe ellips, op basis van een horizontale en een verticale straal. Vergeet niet om het middelpunt met de argumenten `cx` en `cy` te bepalen.

4.5 Veelhoek

Syntax

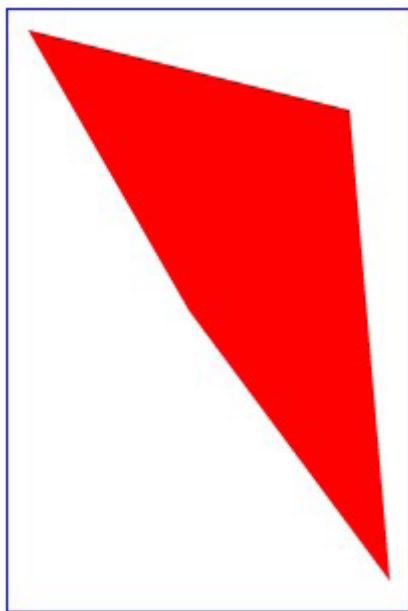
```
<polygon points="x1,y1 x2,y2 x3,y3 x4,y4" style="stijlopties">
```

x1	De x-coördinaat van het eerste punt van de veelhoek.
y1	De y-coördinaat van het eerste punt van de veelhoek.
x2	De x-coördinaat van het tweede punt van de veelhoek.
y2	De y-coördinaat van het tweede punt van de veelhoek.
x3	De x-coördinaat van het derde punt van de veelhoek.
y3	De y-coördinaat van het derde punt van de veelhoek.

x4	De x-coördinaat van het vierde punt van de veelhoek.
y4	De y-coördinaat van het vierde punt van de veelhoek.
style	De stijlkenmerken in css.

Voorbeeld

```
<polygon points="10,10 90,150 190,285 170,50"
           style="fill:red;">
```



4.5_ellips.html

De bovenstaande rode veelhoek bezit vier hoekpunten.

4.6 Veelvoudige lijn

Syntax

```
<polyline points="x1,y1 x2,y2 x3,y3" style="stijlopties">
```

Met `<polyline>` kan je een lijn trekken die uit verschillende delen bestaat. Daarbij kan je zoveel punten toevoegen als je zelf wil. Je laat elk coördinaat bij elkaar horen door de beide waarden met een komma te scheiden, zoals bij `x1, y1` of `x2, y2`.

x1	De x-coördinaat van het eerste punt van de lijn.
y1	De y-coördinaat van het eerste punt van de lijn.
x2	De x-coördinaat van het tweede punt van de lijn.
y2	De y-coördinaat van het tweede punt van de lijn.
x3	De x-coördinaat van het derde punt van de lijn.
y3	De y-coördinaat van het derde punt van de lijn.
x4	De x-coördinaat van het vierde punt van de lijn.
y4	De y-coördinaat van het vierde punt van de lijn.
style	De stijlkenmerken in css.



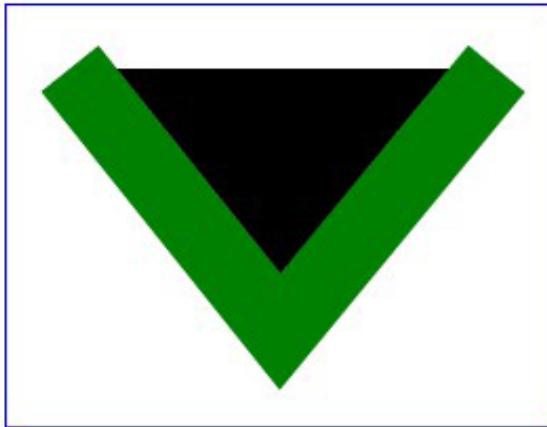
The advertisement features a large white Vestas wind turbine against a blue sky with wispy clouds. The Vestas logo is in the top right corner. The main headline reads "Transform the forces of wind". Below it, the tagline is "Join Vestas and innovate wind technology". A descriptive paragraph at the bottom left discusses job opportunities at Vestas, mentioning over 20,000 colleagues and a focus on reducing CO₂ emissions. The bottom of the ad includes the slogan "Wind. It means the world to us."™



Click on the ad to read more

Voorbeeld

```
<polyline points="30,30 130,155 233,30"
           style="stroke:green;stroke-width:35px;">
```

**4.6_veelvoudigeLijn.html**

Met één enkele opdracht teken je een v-symbool. Je bepaalt het punt linksboven, 30, 30, daarna het onderste punt, 130, 155, om af te sluiten met het punt rechtsboven, 233, 30. Je merkt ongetwijfeld op dat er tussen de benen van de v-figuur een zwart vlak verschenen is. Dat is de standaard vulkleur. Die kan je nog aanpassen naar wit met de eigenschap fill.

```
<polyline points="30,30 130,155 233,30"
           style="stroke:green;stroke-width:35px;fill:white;">
```

**4.6_veelvoudigeLijnWit.html**

Css of svg?

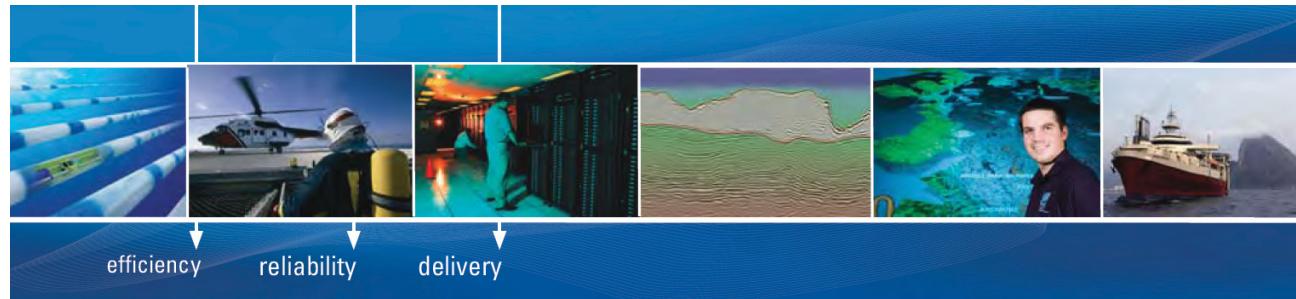
Naast de css-eigenschappen in het argument `style` kan je ook argumenten van `svg` zelf gebruiken, zoals `stroke`, `fill`, ... In het voorbeeld hieronder gebruik je die `svg`-eigen argumenten als volwaardige argumenten binnen de `svg`-tag.

```
<rect width="10" height="10" stroke="#BBB"
      fill="rgb(125,30,75)">
```

Echter kan je al die opmaakargumenten ook rechtstreeks in de `style` stoppen, alsof het onvervalste css-eigenschappen zouden zijn.

```
<rect width="10" height="10" style="stroke:#BBB;
      fill:rgb(125,30,75);">
```

Aan jou de keuze met welke techniek je werkt. In dit boek opteren we voor de tweede manier. Zo verminder je namelijk de hoeveelheid argumenten en groepeer je alle opmaakopdrachten onder één argument. Dat bevordert de leesbaarheid van de code.



As a leading technology company in the field of geophysical science, PGS can offer exciting opportunities in offshore seismic exploration.

We are looking for new BSc, MSc and PhD graduates with Geoscience, engineering and other numerate backgrounds to join us.

To learn more our career opportunities, please visit www.pgs.com/careers



Download free eBooks at bookboon.com

5 Paden

Net zoals bij de canvas kan je ook binnen <svg>-containers met paden werken. Paden zorgen ervoor dat een figuur als één gesloten geheel beschouwd wordt. De werkwijze tussen canvas en svg is echter wel heel verschillend. In svg reik je met kapitalen of onderkasten binnen één argument opdrachten aan. Een ongewone werkwijze, toch wen je er snel aan. Een kapitaal gebruik je voor absolute verwijzingen, met onderkasten maak je relatieve verwijzingen. Een kleine greep uit de mogelijkheden.

5.1 move to

Syntax

Met de letter M zet je de pen klaar op het werkveld.

```
<path d="Mx y Z">
```

Het argument d – data – geeft de instructies voor het pad weer.

M Voer een move to-opdracht uit.

x De x-coördinaat van het punt waarnaar de pen verplaatst moet worden.

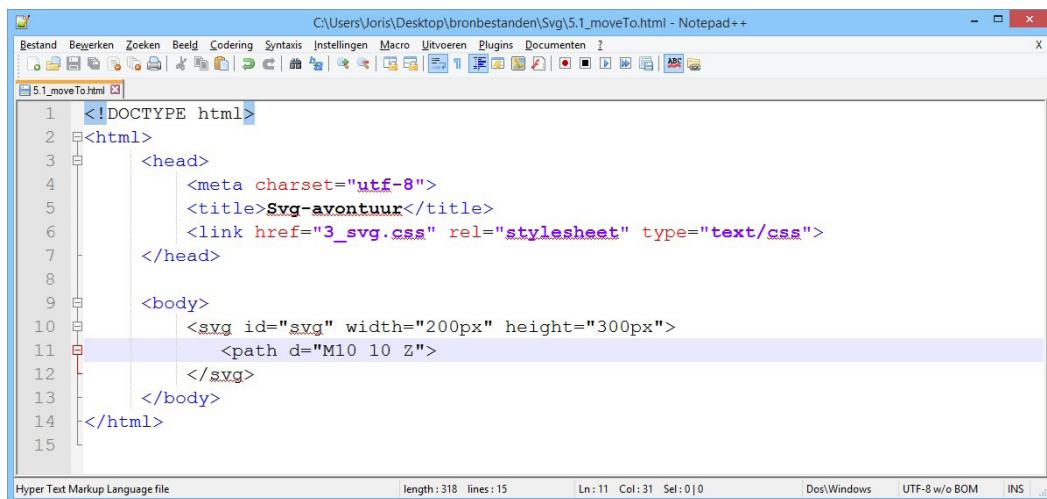
y De y-coördinaat van het punt waarnaar de pen verplaatst moet worden.

Z Met Z sluit je een pad af.

Merk je bovendien op dat je tussen de M en de x-coördinaat geen spatie laat?

Voorbeeld

```
<path d="M10 10 Z">
```



The screenshot shows a Notepad++ window with the file '5.1_moveTo.html'. The code is as follows:

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>Svg-avontuur</title>
6     <link href="3_svg.css" rel="stylesheet" type="text/css">
7   </head>
8
9   <body>
10    <svg id="svg" width="200px" height="300px">
11      <path d="M10 10 Z">
12    </svg>
13  </body>
14</html>
15

```

The line containing the path instruction is highlighted.

5.1_moveTo.html

Met de opdracht M binnen het argument d zet je de pen om te tekenen op coördinaat 10,10 klaar. Met Z sluit je een pad af.

5.2 line to

Syntax

Vervolgens kan je een lijn binnen dat pad tekenen. Dat bereik je met de letter L. Plaats geen spatie tussen de L en de x-coördinaat.

```
<path d="Mx y Lx1 y1 Z">
```

Het argument d geeft de instructies voor het pad weer.

M Voer een move to uit.

x De x-coördinaat van het punt waarnaar de pen verplaatst moet worden.

y De y-coördinaat van het punt waarnaar de pen verplaatst moet worden.

L Voer een line to uit.

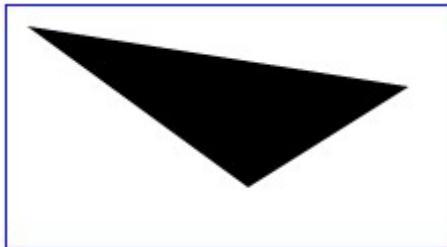
x1 De x-coördinaat van het eindpunt van de lijn.

y1 De y-coördinaat van het eindpunt van de lijn.

Z Met Z sluit je een pad af.

Voorbeeld

```
<path d="M10 10 L200 40 L120 90 Z">
```



[5.2_lineTo.html](#)

Met deze opdracht zet je de pen om te tekenen klaar op coördinaat 10,10. Daarna teken je de lijn naar punt 200,40 om tot slot te eindigen bij coördinaat 120,90. Daarna sluit je het pad met de kapitaal **Z** af. Onthoud wel dat je een pad uit minimaal drie punten laat bestaan. Alleen zo kan een pad ook daadwerkelijk een gesloten geheel vormen.

5.3 Andere

Er zijn nog mogelijkheden om paden te bouwen. Zo kan je andere letters toepassen, zoals **C** voor béziercurve of een **A** voor een elliptische boog.

nrc carrière >
samen ambities waarmaken

nrcarriere.nl
Ook voor starters

nrcarriere.nl
Ook voor starters

Download free eBooks at bookboon.com

136

 Click on the ad to read more

6 Teksten

Naast figuren kan je ook teksten binnen een <svg>-container plaatsen. Net zoals bij de canvas schrijf je geen tekst, maar teken je lettervormen binnen de svg. Verder kan je tekst indelen en hyperlinks creëren.

6.1 Tekst tekenen

Syntax

Tekst plaats je tussen een <text>-container. Bepaal voor tekst de startpositie.

```
<text x="getal" y="getal" style="stijlopties"></text>
```

x De x-coördinaat van het beginpunt van de tekst.

y De y-coördinaat van het beginpunt van de tekst.

style De stijlkenmerken in css.

Voorbeeld

```
<text x="30" y="20" style="font-family:Verdana;">  
Vectoriële tekst  
</text>
```



[6.1_tekstTekenен.html](#)

Met de bovenstaande code zet je de tekst Vectoriële tekst positioneer je met een x-waarde en een y-waarde. op het <svg>-vlak.

6.2 Tekst indelen

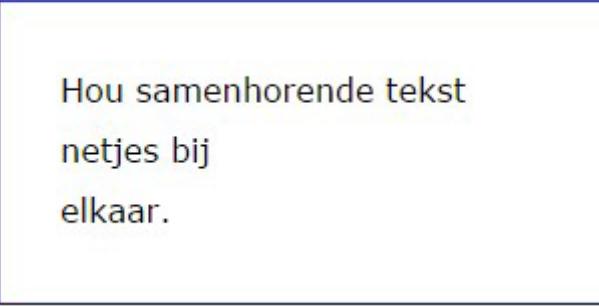
Syntax

Wil je één tekst opsplitsen in diverse lijnen of posities? Dan kan je dat handmatig realiseren door meerdere `<text>`-containers te bouwen – zoals in het vorige voorbeeld – ofwel gebruik je `<tspan>`.

```
<tspan x="getal" y="getal">Tekst</tspan>
```

Voorbeeld

```
<text x="30" y="50" style="font-family:Verdana;">  
    Hou samenhangende tekst  
    <tspan x="30" y="80">netjes bij</tspan>  
    <tspan x="30" y="110">elkaar.</tspan>  
</text>
```



Hou samenhangende tekst
netjes bij
elkaar.

6.2_tspan.html

Binnen een `<text>`-container kan je één of meerdere `<tspan>`-containers plaatsen. Met die container kan je stukken tekst apart positioneren. Een `<tspan>` bezit namelijk eigen coördinaten, maar blijft tot één `<text>`-container behoren.

7 Animatie

Net zoals de canvas kan je ook een svg-illustratie animeren. Daarvoor kan je JavaScript gebruiken, waarmee je in de svg-dom-structuur ingrijpen. Ofwel kies je voor zuivere svg-animatie-opdrachten. We bieden hieronder twee voorbeelden van zulke svg-instructies.

7.1 Transparantie

Syntax

Wil je een figuur transparant maken tijdens een animatie? Dan helpt `animate` je op weg.

```
<animate attributeType="type" attributeName="soort"  
repeatCount="herhaling">
```

`attributeType` De taal die je voor de animatie gebruikt.

`attributeName` Het soort effect dat je wenst toe te passen.

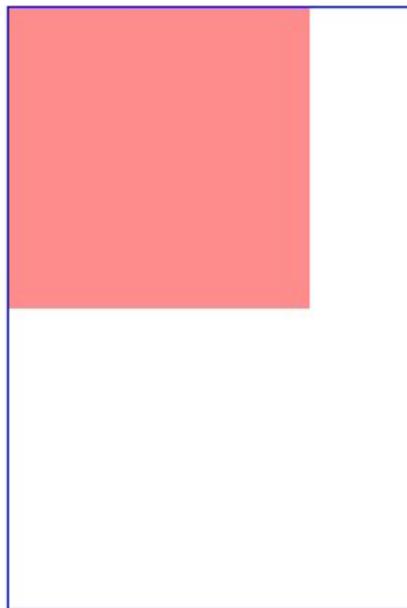
`repeatCount` Hoe vaak de animatie herhaald moet worden.

Afhankelijk van het effect kunnen er nog andere argumenten toegevoegd worden.



Voorbeeld

```
<rect width="150" height="150" style="fill:red;">
  <animate attributeType="CSS" attributeName="opacity" from="1"
    to="0" dur="5s" repeatCount="indefinite">
</rect>
```

**7.1_transparantie.html**

In de bovenstaande code maak je een rechthoek op. Binnen de container van die rechthoek plaats je vervolgens een effect met css (attributeType). Dat effect is hier een overgang in transparantie (attributeName) tussen waarde 1, ofwel volledig dekkende kleur (from) en waarde 0 (to), ofwel volledig doorzichtig. Die overgang neemt vijf seconden in beslag (dur). Met indefinite geef je aan dat de animatie onbeperkt blijft lopen.

7.2 Vervormen

Syntax

De animateTransform kan je gebruiken voor animaties waarbij je het object vergroot of verkleint, scheef trekt of verplaatst.

```
<animateTransform attributeType="type" attributeName="soort"
    type="" repeatCount="herhaling">
```

attributeType De taal die je voor de animatie gebruikt.

attributeName Het soort effect dat je wenst toe te passen.

type Het individuele type van effect dat je gebruikt.

repeatCount Hoe vaak de animatie herhaald moet worden.

Afhankelijk van het effect kunnen er nog andere argumenten toegevoegd worden.

Success here is about overcoming challenges – especially the ones we didn't even think of. I look forward to that.

- Scott, Mechanical Engineering Group Leader

Produce More. Conserve More. Improve Lives.

Monsanto has always embraced innovation and always focused on helping to make a better world. You can see it in our groundbreaking products and in our dynamic environment where your skills and your career can grow and develop. We know that every day, new ideas can come from anyone, anywhere. At Monsanto, you'll be respected, you'll contribute to the bottom line and you'll help farmers feed the world.

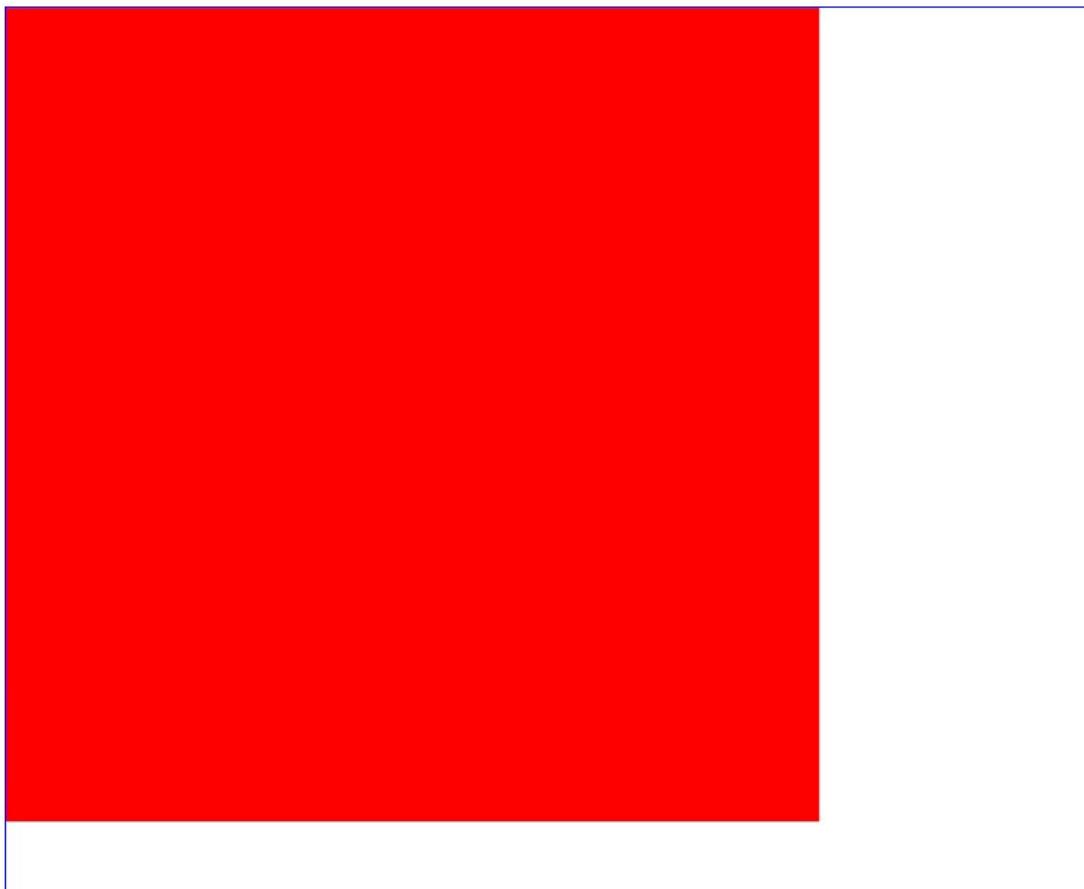
Start right now: www.monsanto.com/students

Monsanto is an equal opportunity employer, we value a diverse combination of ideas, perspectives and cultures.
EEO/AA EMPLOYER M/F/D/V © 2013 Monsanto Company

MONSANTO

Voorbeeld

```
<rect width="5" height="5" style="fill:red;">
  <animateTransform attributeName="transform" type="scale" from="0 0" to="600 600"
    dur="15s" repeatCount="freeze">
</rect>
```



7.2_vervormen.html

Wanneer je aan het vervormen slaat, gebruik je als attributeType geen css maar wel zuivere xml. Vervolgens geef je de verzamelnaam van de effecten bij attributeName op, namelijk transform. Daarna kan je het specifieke effect bij type aanduiden, in dit geval scale. Bij from en to geef je het bereik op waarbinnen geschaald moet worden. De herhaling staat hier op een stop (freeze). Daardoor toont de animatie één keer, waarna het eindresultaat op het scherm blijft staan.

8 Oefenen

Om het tekenen met code stevig onder de knie te krijgen, is het belangrijk dat je veel oefent en experimenteert. Start met eenvoudige tekeningen zoals een doosje of een fabriekje. Je kan ook tekenen in Inkscape en daarna de code bekijken.

Onthoud daarbij dat dé oplossing niet bestaat. Je kan een vierkant bijvoorbeeld zowel als vier lijnen tekenen, als met de rechthoekmethode.

Op claimjekennis.soopbooks.com zie je verschillende opgaven staan. Echter kan je met je code vast lopen. Gebruik de element inspector van je browser – kijk bijvoorbeeld de console bijvoorbeeld na op fouten – en spoor zo problemen op. Gebruik bovendien ook de volgende checklist.

8.1 Checklist

- Html-containers: heb je alle html-containers correct en op de juiste plek afgesloten?
- Svg-containers: heb je alle svg-containers correct en op de juiste plek afgesloten?
- Puntkomma: sluit elke css-instructie met een puntkomma af.
- Aanhalingstekens: gebruik je de juiste rechte aanhalingstekens in je code? Sluit je ze ook correct af?

