canvasJS

In het canvas gedeelte van de website ga je aan de slag met het maken van tekeningen via de programmeertaal Javascript.

het principe van canvas is heel simpel; een html element krijgt een hoogte en een breedte en krijgt een x- en een y- as waar je met behulp van javascript code tekeningen op kan maken.

Waar vroeger Flash werd gemaakt om spelletjes en animaties te maken op het web, wordt er tegenwoordig meer en meer gewerkt met Canvas.

Kijk hier voor een kleine impressie over wat met canvas mogelijk is:

http://www.canvasdemos.com/type/games/

# DE SETTING:

<canvas> is in feite een html element, net als een <div> of een <h1>. Het bijzondere van <canvas> is dat je met behulp van javascript extra eigenschappen kan toevoegen aan dit element, zodat je er met behulp van scripttaal op kunt tekenen. Vervolgens, als je wat gevorderd bent, kun je deze tekeningen laten animeren en interactief maken, en voor je het weet heb je een game gemaakt!

# STRUCTUUR:

Javascript wordt geschreven in een .js bestand. In dit project vind je deze bestanden in de folder 'scripts'. In canvas.js gaan we aan de slag tijdens dit gedeelte.

op regel 1 en regel 6 staat al wat code. Deze code zegt simpelweg: alles wat hier tussen in komt te staan, bestaat alleen hier tussen, en nergens anders. Op deze manier wordt onze code niet overschreven door andere scripts en vice versa.

het daadwerkelijke canvas element staat ergens in het index.html bestand. Dit is het bestand dat we zien als we de pagina openen in een webbrowser.

canvas.js is de enige plek waar we code gaan aanpassen tijdens dit gedeelte.

# OPDRACHTEN:

**1. Op welke regel is het canvas element in de html gezet?**

**2. Open de index.html en scroll naar het canvas gedeelte.**

**3. Ga in de HTML op zoek naar de regel ‘<div class = “section section-canvas”>’. Zet na ‘section-canvas’ maar voor de dubbele quote het woord ‘active’.**

Dit zorgt er voor dat als je de browser opent, je niet steeds naar de canvas sectie hoeft te scrollen.

**4. Open in sublime het bestand 'canvas.js'.**

## het <canvas> element gebruiksklaar maken

Aangezien we het echte scripten in javascript gaan doen, zal ik in javascript code eerst moeten aangeven wat mijn canvas element is waar ik op wil tekenen.

Zoals je kunt zien in de index.html, heb ik de canvas een id 'attribuut' meegegeven. Dit is een unieke naam die ik heb gegeven aan het element, zodat ik in Javascript dit element kan aanroepen.

**5. voeg in regel 3 de volgende regel code toe:**



Met deze regel javascript zeg je eigenlijk het volgende: “Browser, reserveer voor mij een mini klein plekje in het computergeheugen en noem het ‘canvas’. (dit doe je met de var ‘keyword’) Zet in dit plekje het html element wat ik als canvas wil gebruiken. (document.getElementById(‘canvas’).

Je hebt nu een variabele gemaakt. Daarvan zul je er in je loopbaan nog miljoenen van gaan maken ☺

**6. voeg in regel 4 de volgende regel code toe:**



Met deze regel code maak je weer een nieuwe variabele, genaamd ‘ctx’. (staat voor context). In ctx zet je nu het canvas element wat je net in het geheugen hebt gezet, en je activeert de canvas ‘superkracht’ met getContext(‘2d’). Vanaf nu is het canvas element een tekenbaar element geworden!

Je canvas ligt voor je klaar, tijd om te gaan tekenen!

Het is erg handig als je ook even in het computergeheugen vastlegt wat eigenlijk de afmetingen zijn van jouw canvas. Hoe breed en hoe hoog is deze eigenlijk?

**7. Voeg in regel 6 en 7 de volgende code toe:**



Hier maak je weer 2 variabelen aan. In de ene zet je de breedte van het canvas element, en in het andere zet je de hoogte van het canvas element.

## debuggen

Soms kunnen er wel eens foutjes in je code sluipen. Daarom is het verstandig om zo af en toe eens te kijken wat er in je variabelen zit. In de browser zit een ‘console’ die je kunt openen met ‘f12’. Klik vervolgens op ‘console’. Hier vind je (fout)meldingen die door javascript worden gegenereerd.

Je kunt hier gebruik van maken door ergens in je code de volgende regel te schrijven:

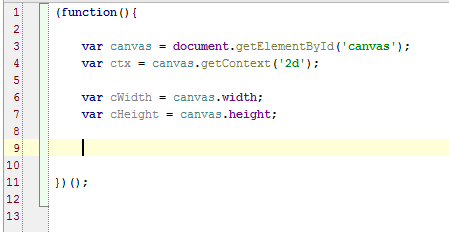


Je logt nu de inhoud van de variabele ‘cWidth’ naar de console.

**8. log de variabele cWidth naar de console en maak een printscreen van wat je ziet:**

**9. Zet de console.log() regel die je net hebt beschreven eens op regel 3. Wat voor melding krijg je nu, en waarom krijg je deze melding?**

Tijd om te gaan tekenen nu. Als het goed is heb je nu deze code in canvas.js staan, en geen foutmeldingen in de browserconsole:



Om te tekenen hebben we een heel aantal verschillende functies (tools) die we kunnen gebruiken. We kunnen vormen tekenen, maar ook simple lijnen (paths). Dat is waar we mee gaan beginnen.

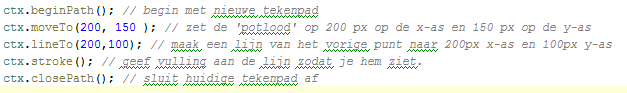
Benodigde functies:

beginPath()  
*met deze functie geven we aan dat we een nieuwe reeks lijnen gaan maken, of één simple lijn.*

moveTo(x, y)  
*met deze functie geven we aan op welke positie we de ‘potlood’ op het canvas willen hebben. De x en de y waarden bepalen de positie.*

lineTo(x, y)  
*met deze functie geven we aan naar welke positive we een lijn willen trekken.*  
stroke()   
*met deze functie geven we aan dat de lijnen moeten worden ingekleurd.*  
closePath()  
met deze functie geven we aan dat we klaar zijn met de gemaakte lijnen.

**10. zet de volgende code vanaf regel 9 in canvas.js**



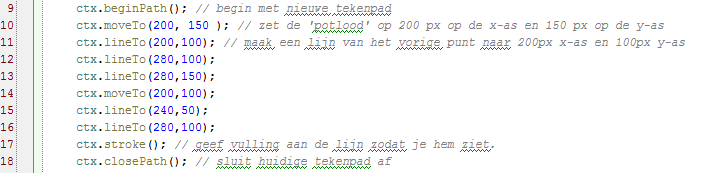
**11. bekijk het resultaat in de index.html**

Trouwens, zie je de ‘//’ die in de code staat? Alles wat daarna komt wordt niet geïnterpreteerd door de computer. Dit wordt gebruikt om commentaar bij de code te zetten, om uitleg te geven bijvoorbeeld.

De eerste lijn is gemaakt! Te gek!

We gaan onze oude code uitbreiden zodat het een bekende vorm wordt.

**12. Breidt de bestaande code uit met onderstaande code**

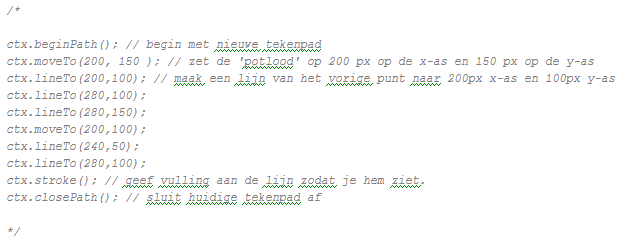


Mooie crib! Aan jou de taak om deze een deur te geven.

**13. Breidt de code zodanig uit dat in het huis een deur komt.**

We hebben nu voornamelijk met de ‘path’ functionaliteiten gewerkt, maar er zijn nog meerdere vormen en patronen die we kunnen gebruiken om onze tekening af te maken.

**14. zet de huidige ‘tekencode’ in commentaar zoals te zien hieronder**



Ook deze tekens (/\* om te beginnen en \*/ om te eindigen) worden gebruikt om commentaar te voorzien. Het grote verschil met commentaar op deze manier is dat je commentaar kunt schrijven van meerdere regels. Zoals je ziet kan het ook gebruikt worden om bestaande code even niet te laten interpreteren door de computer. Ververs maar eens. Je zult zien dat er nu geen tekening staat.

Er zijn ook complexere vormen te maken dan alleen lijntjes als we zojuist hebben gedaan. Eén van dit soort vormen is een boog.

Een boog maak je met behulp van de arc() functionaliteit.

Om een boog te maken heb je iets meer informatie nodig dan alleen een x- en een y-as. Die heb je ook nodig, want je moet weten waar op het canvas de cirkel geplaatst moet worden. Buiten dat, zul je ook de radius van de boog moeten weten, de hoek waarin de boog moet starten, de hoek waarin de boog moet stoppen, en of de boog met de klok mee of tegen de klok in getekend moet worden.

De hoeken worden niet berekend in graden, maar in radialen. Pi, ofwel 3,14 in radialen is de helft van een cirkel. In Javascript schrijven we Pi op deze manier: ‘Math.PI’

**15. begin onder de code die momenteel in commentaar staat, een nieuw path te openen en een boog (arc) te maken:**



*ctx.arc(centrum x-as, centrum y-as, radius, starthoek, eindhoek)*

Zoals aangegeven is Pi, 3,14 de helft van een cirkel in radialen.

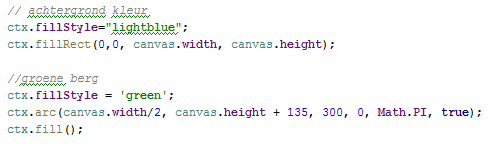
**16. Bedenk hoe je van deze boog een cirkel kunt maken en teken deze op het canvas**

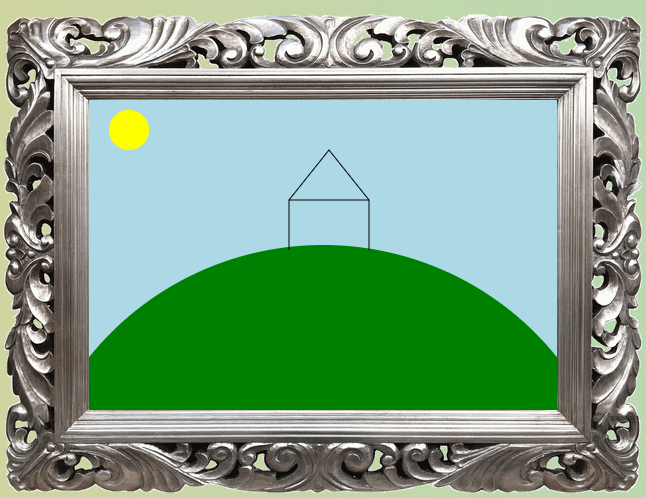
We hebben nu voornamelijk met de stroke() functie gewerkt om lijnen om onze vormen te trekken. We kunnen ook de fill() functie gebruiken om in plaats van een lijn, een opvulling te krijgen.

**17. verander de laatste ctx.stroke() in je code naar ctx.fill() en bekijk het resultaat.**

**18. laten we het kunstwerk wat opleuken met een achtergrondje en een mooie groene berg. Als je de afmetingen hebt gelaten zoals in de opdrachten, gaat het allemaal goed uitkomen. Zo niet zul je zelf wat met de waarden moeten spelen. Zet de commentaar voor de code voor het huis weer uit, zodat deze weer zichtbaar wordt.**

**Boven de code van het huis zet je onderstaande code:**





**19. bij het maken van de berg heb ik het over canvas.width / 2. Waarom zouden we dit als positie gebruiken?**

**20. als laatste waarde aan de arc geven we true mee. Waarom doen we dit?**

**21. Voor de achtergrond hebben we gebruik gemaakt van de functie fillRect(). Wat zou deze functie doen?**

Laten we een paadje aan leggen richting het huis. Hiervoor gaan we een soort bocht gebruiken die veel flexibeler is dan de arc() functie, namelijk ‘quadradic curves’. Met deze curves kun je aangeven hoe scherp bepaalde bochten dienen te zijn door middle van een control point.

Zie onderstaande link voor een duidelijk voorbeeld.

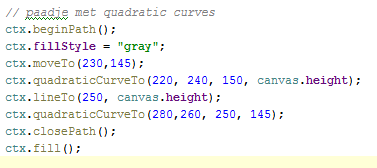
<http://www.rgraph.net/blog/2013/january/an-example-of-the-html5-canvas-quadraticcurveto-function.html>

naast een quadratic curve bestaat er ook een Bezier curve, waarbij niet één maar twee control points aangegeven worden, voor nog meer mogelijkheden en flexibiliteit in je vormen. Deze wordt echter al heel snel erg ingewikkeld en passen we voor dit projectdeel dan ook niet toe.

<http://www.rgraph.net/blog/2013/january/an-example-of-the-html5-canvas-beziercurveto-function.html>

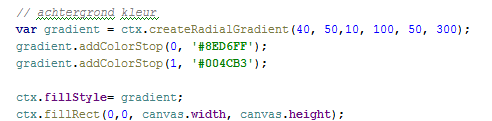
Quadratic curves maken we met de quadraticCurveTo() functie.

**22. schrijf onderstaande code over onder in je script.**



**23. Geef in eigen woorden aan welke waarden voor welke doeleinden zijn van de quadraticCurveTo() functie. (er zijn 4 waarden die moeten worden meegegeven, waar staan die voor?).**

**24. Maak van de achtergrond in plaats van de lineaire blauwe kleur een verloopje in de kleur:**



**25. in de addColorStop maken we gebruik van een vreemde tekenset (‘#8ED6FF’) wat is dit?**

**26. Het is nu aan jou om de laatste details aan jou prachtige kunststuk te hangen. Denk bijvoorbeeld aan een wolkje, een smiley in de zon, een auto voor de deur, een vogel op het dak, een schoorsteen, ramen, noem maar op!**

**27. Zoek op wie Bob Ross was. Deze man heeft jou niks meer te leren!**

[**https://www.youtube.com/watch?v=pw5ETGiiBRg**](https://www.youtube.com/watch?v=pw5ETGiiBRg)