

No plan survives first contact with the enemy. (Helmuth von Moltke)

## 1 Inleiding

P5.js is een bibliotheek (library) met functionaliteit waarmee je snel en eenvoudig tekeningen en animaties kunt maken voor het web. Misschien heb je wel gehoord van Processing.js maar dat is eigenlijk een voorloper van P5.js. De taal die je voor P5.js gebruikt is gewoon javascript. Bij Processing is de basistaal Java. Wat het zo fijn maakt is dat je met simpele commando's figuren kunt tekenen. Zo kun je een rechthoek tekenen als volgt:

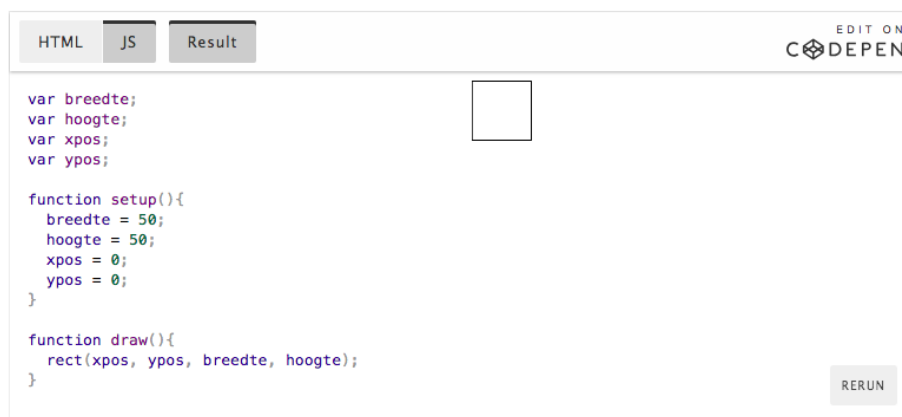
```
rect(20, 30, 40, 20);  
//Rechthoek op positie (20, 30), 40 pixels breed en 20 pixels hoog.
```

### 1.1 Canvas

Net als bij het schilderen teken je op een canvas, een deel van je browserwindow. Je kunt zelf bepalen hoe groot of hoe klein het canvas is maar alles wat je tekent moet op het canvas komen anders zie je het niet (maar het is er wel). Zoals je gewend bent zit het punt (0, 0) helemaal linksboven op je canvas. Alle afmetingen die je gaat gebruiken zijn in pixels.

### 1.2 Draw loop en setup

De basis van je tekening of animatie is altijd de draw loop. De code in deze loop blijft achter elkaar uitgevoerd worden. Een eerste voorbeeldprogramma is als volgt:



<https://codepen.io/jurjenh/pen/gGPMLY>

Dit programma tekent een rechthoek en blijft dat 60 keer per seconde doen. De framerate is namelijk 60 fps (frames per second). Wat je dus zult zien is een rechthoek en meer niet. Wil je dat de code één keer wordt uitgevoerd of stopt met herhalen dan gebruik je:

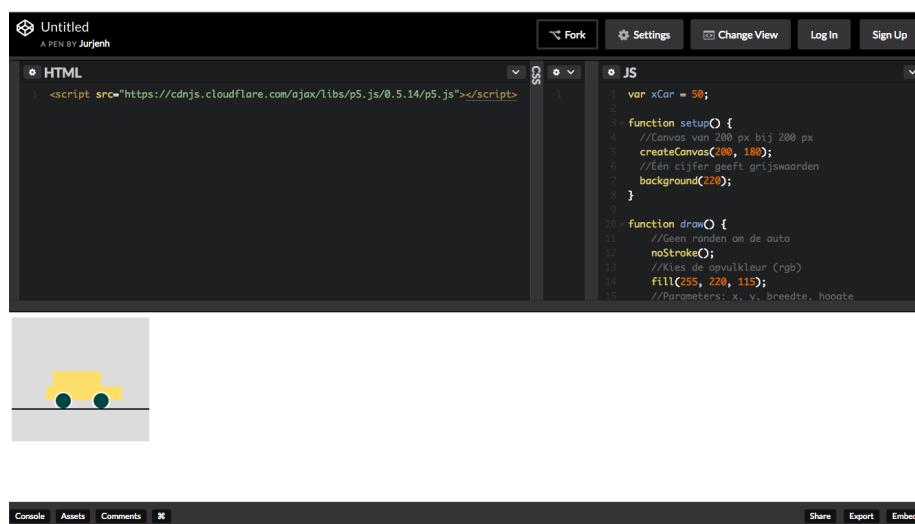
```
noLoop();
```

Je ziet ook de setup-functie. Deze functie wordt één keer uitgevoerd aan het begin van je programma.

### 1.3 Codepen

Codepen.io is een omgeving waar je makkelijk kunt experimenteren met HTML, CSS en JS. Als je een (gratis) account aanmaakt kun je je probeersels ook bewaren. Hier maken we gebruik van *Codepen's* gemaakt door een informaticadocent van Metis Montessori Lyceum. Steeds is de HTML heel simpel: er wordt alleen maar een bibliotheek geladen via

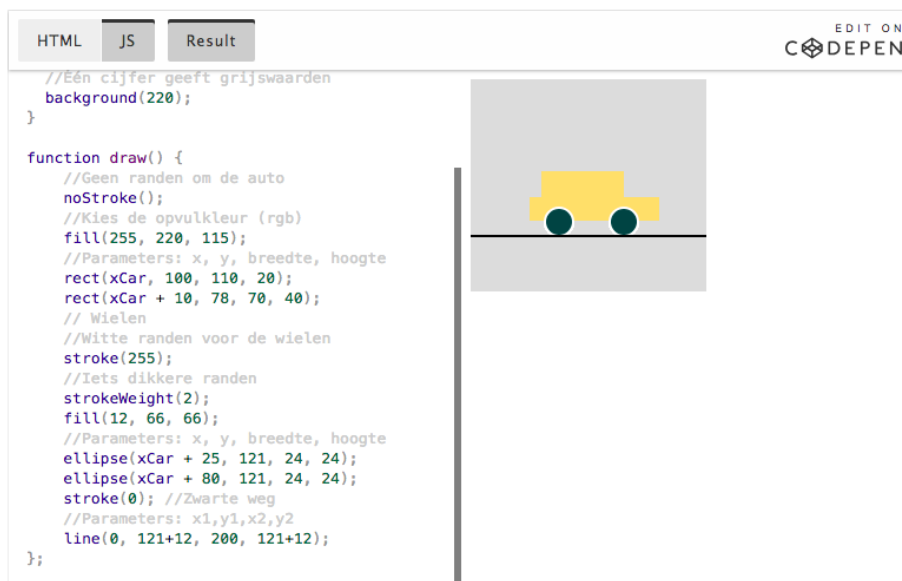
```
<script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.5.14/p5.js"></script>
```



## 2 Je eerste tekening

Je gaat aan het einde van deze les je eerste tekening maken. Gebruik zo veel mogelijk variabelen om de verschillende lengtes aan te geven. Je zult merken dat je tekening dan veel sneller te veranderen is zonder dat je moet gaan zoeken waar elk getal ook alweer voor staat.

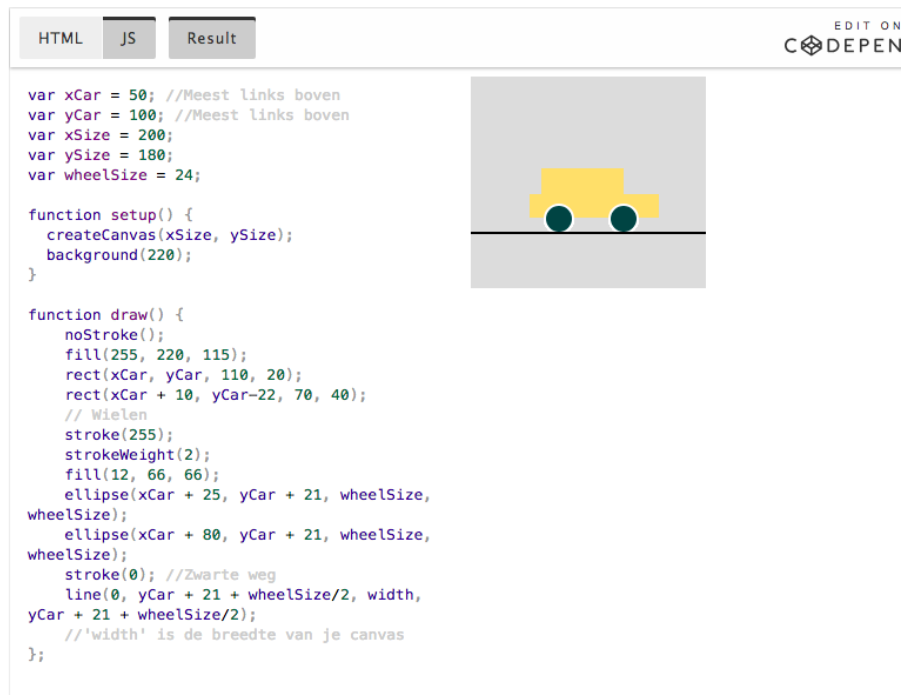
Een aantal belangrijke figuren zijn ellipsen, rechthoeken en lijnen. Je kunt ze vullen met een bepaalde kleur of de rand uitzetten of van kleur veranderen. Kijk maar eens naar het voorbeeld hieronder.



<https://codepen.io/jurjenh/pen/boEgrp>

**Opdracht 2.1.** *Open de codepen en onderzoek wat elke regel in de draw functie doet. Waarom is de auto geel? Wat is de kleur van de banden? Waarom worden er twee rechthoeken getekend? En waarom twee ellipsen? Voeg tussen de twee regels voor de rechthoeken een regel `fill(255,0,0);` toe. Wat verandert? Waarom?*

Zoals je ziet zijn al die getallen wel onoverzichtelijk. Wat als we nu iets willen veranderen? Laten we hetzelfde voorbeeld nu eens maken met behulp van variabelen.



<https://codepen.io/jurjenh/pen/BwjpML>

**Opdracht 2.2.** Ga voor elke variabele na hoe er gebruik van wordt gemaakt. Wat gebeurt er als je bovenaan de `wheelSize` verandert in 10? Of in 60? Hoe kun je zorgen voor een breder canvas?

Dat ziet er al beter uit. Natuurlijk kunnen we ook nog een variabele toevoegen voor bepaalde kleuren of de hoogte van de wielen en meer. We gaan dat nu niet doen.

## 2.1 Je eigen tekening

Ga nu zelf een tekening maken. Je mag zelf weten wat je tekent. Zorg dat je oefent met de verschillende vormen en kleuren etcetera. Je hebt hiervoor een html pagina nodig met het volgende erin:

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title></title>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.5.14/p5.js"></script>
    <!-- het javascript bestand hieronder bevat je tekening -->
    <script src="tekening.js"></script>
  </head>
  <body>
  </body>
</html>

```

Noem deze file `index.html`. Als je hem in een browser opent zul je een lege pagina zien. Dat verandert zodra je in dezelfde map een tweede file `tekening.js` aanmaakt met daarin instructies zoals in de voorbeelden hiervoor.

**Opdracht 2.3.** *Maak een map `P5tekening` aan en bewaar daarin twee bestanden:*

1. *`index.html` met de HTML zoals boven*
2. *`tekening.js` volgens de structuur van de voorbeelden met twee functies: `setup()` en `draw()`*

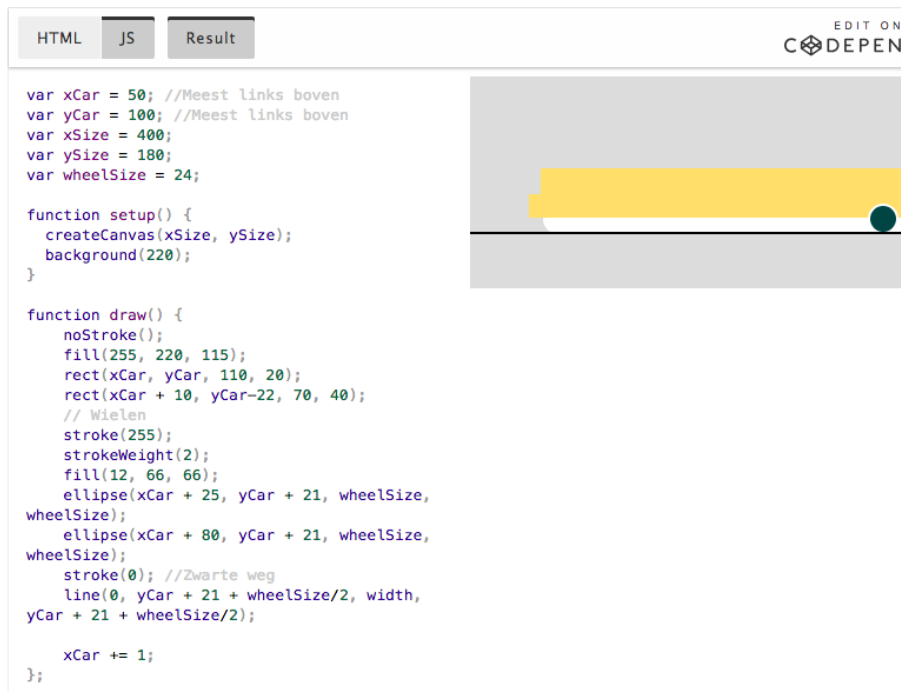
*Zorg dat de achtergrond van je canvas niet wit is, en teken een aantal lijnen en vormen waarbij je varieert in kleur en dikte.*

**Opdracht 2.4. BONUS** *Verplaats de twee bestanden uit vorige opdracht naar een repo `P5tekening` en maak je werk zichtbaar via `username.github.io/P5tekening`. De uitleg hoe zo'n repo een publieke website wordt vind je hier als tekst (<https://help.github.com/articles/configuring-a-publishing-source-for-github-pages/>) en hier uitgelegd door Daniel Shiffman <https://youtu.be/bFVtrlyH-kc>.*

Als je wilt weten wat je nog meer kunt gebruiken, en dat is heel veel, kijk dan op <https://p5js.org/reference/>.

### 3 Animaties

Als je variabelen hebt gebruikt zoals in het voorbeeld is het eenvoudig om er een animatie van te maken! We voegen nu namelijk n regel code toe. Deze regel code verhoogt xCar met 1, iedere keer dat de draw loop doorlopen wordt. Zo wordt er steeds een nieuwe auto getekend. Alleen jammer dat de oude auto ook blijft staan.



```
HTML JS Result EDIT ON CODEPEN

var xCar = 50; //Meest links boven
var yCar = 100; //Meest links boven
var xSize = 400;
var ySize = 180;
var wheelSize = 24;

function setup() {
  createCanvas(xSize, ySize);
  background(220);
}

function draw() {
  noStroke();
  fill(255, 220, 115);
  rect(xCar, yCar, 110, 20);
  rect(xCar + 10, yCar-22, 70, 40);
  // Wielen
  stroke(255);
  strokeWeight(2);
  fill(12, 66, 66);
  ellipse(xCar + 25, yCar + 21, wheelSize,
wheelSize);
  ellipse(xCar + 80, yCar + 21, wheelSize,
wheelSize);
  stroke(0); //Zwarte weg
  line(0, yCar + 21 + wheelSize/2, width,
yCar + 21 + wheelSize/2);

  xCar += 1;
};
```

<https://codepen.io/jurjenh/pen/qPbrbe>

Wat gebeurt er nou? Elke keer wordt in de draw loop alles getekend over de tekening die er al stond. Als we dus een nieuwe auto willen zullen we de achtergrond steeds opnieuw moeten tekenen. Bovendien rijdt de auto uit het scherm. Dat gaan we later oplossen.

**Opdracht 3.1.** *Probeer te bedenken wat er moet gebeuren om de uittrekkende auto te veranderen in en bewegende auto. Hint: hoe kun je de auto laten verdwijnen? Nog een hint: je hoeft maar een regel code te verplaatsen.*

Nu we steeds meer code gaat toevoegen is het handig om een *functie* te maken voor onze auto. We voeren wat wijzigingen door en krijgen dan het volgende voorbeeld. Een rijdende auto!

HTML

JS

Result

EDIT ON  
**CODEPEN**

```

var xCar;
var yCar;
var xSize = 400;
var ySize = 180;
var wheelSize;

function setup() {
  createCanvas(xSize, ySize);
  xCar = 50;
  yCar = 100;
  wheelSize = 24;
}

function draw() {
  background(220); //Één cijfer geeft
  grijswaarden
  drawCar(xCar, yCar, wheelSize);
  xCar += 1;
};

function drawCar(xCar, yCar, wheelSize){
  noStroke();
  fill(255, 220, 115);
  rect(xCar, yCar, 110, 20);
  rect(xCar + 10, yCar-22, 70, 40);
  // Wielen
  stroke(255);
  strokeWeight(2);
  fill(12, 66, 66);
  ellipse(xCar + 25, yCar + 21, wheelSize,
wheelSize);
  ellipse(xCar + 80, yCar + 21, wheelSize,
wheelSize);
  stroke(0);
  line(0, yCar + 21 + wheelSize/2, width,
yCar + 21 + wheelSize/2);
}

```

RERUN

<https://codepen.io/jurjenh/pen/KXVWNR>

**Opdracht 3.2.** Bestudeer de code van de laatste Codepen goed. Voorspel wat er gebeurt bij de volgende aanpassingen en test het uit:

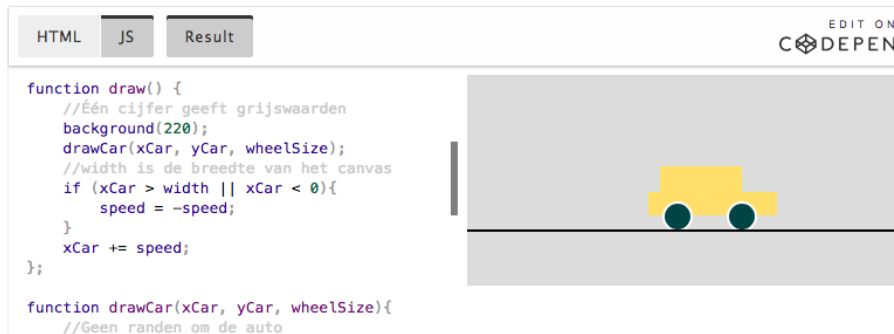
1. In draw: verander `xCar += 1`; in `xCar -= 1`;
2. In draw: verander `xCar += 1`; in `xCar += 5`;
3. In draw: verander `xCar += 1`; in `yCar -= 1`;
4. In drawCar: verander `noStroke()`; in `stroke(0)`;
5. In drawCar: verander `noStroke()`; in `stroke(255)`;
6. In drawCar: verander `noStroke()`; in `stroke(255, 0, 0)`;
7. In drawCar: verander `noStroke()`; in `stroke(0, 255, 0)`;
8. In drawCar: verander `noStroke()`; in `stroke(0, 0, 255)`;

**Opdracht 3.3.** Voeg een regel code toe aan de drawCar functie om de auto een raampje te geven.

**Opdracht 3.4.** Ga terug naar je tekening uit de vorige paragraaf en breng daar beweging in!

## 4 Randen detecteren

Het is wel jammer dat de auto weggrijdt. Kunnen we hem niet laten omkeren? Jazeker! We laten de auto rijden door steeds `xCar` op te hogen met 1. Op een gegeven moment is `xCar` groter dan de breedte van je scherm en is je auto niet meer te zien. Wat we dus doen om dit op te lossen is de snelheid van de auto *om te draaien* als `xCar` te groot of te klein is.



<https://codepen.io/jurjenh/pen/VMaXjd>

Het is nog niet perfect want de auto rijdt eerst helemaal het scherm uit en komt daarna weer terug. Je kunt vast een manier bedenken om dit op te lossen.

**Opdracht 4.1.** *Hoe moet je de code aanpassen om de auto al te laten keren wanneer de voorbumper de rand raakt?*

**Opdracht 4.2.** *Hoe moet je de code aanpassen om de auto nadat die rechts uit beeld is gereden, links weer tevoorschijn komt?*

**Opdracht 4.3.** *Maak alsnog opdracht 2.4 op bladzijde 5.*