Chart.js iteraties

Inhoudsopgave

Inleiding

In onze app willen wij data uit een MongoDB database omzetten in een grafiek. Uit mijn onderzoek bleek dat Chart.js hier een uitstekende library voor is. Om te kijken of de library echt voldoet heb ik in deze POC getest of alles goed werkt. Ook heb ik verschillende iteraties van grafieken gemaakt om te kijken welke het beste bij de data past en welke kleuren en vormgeving het beste zijn.

Data omzetten in een grafiek

Met Chart.js kan je Json data omzetten in een grafiek. Je kunt deze data uit een database halen maar om dit eerst te testen kan je deze data ook eerst in de HTML zetten. Dit ziet er zo uit.

```
var jsonfile = {    //Dit is de json data, deze wordt uiteindelijk uit de database gehaald
    "jsonarray": [{
        "month": "Jan",
        "amount": 100
    }, {
        "month": "Feb",
        "amount": 200
    }, {
        "month": "Maart",
        "amount": 364
    }, {
        "month": "April",
        "amount": 872
    }, {
        "month": "Mei",
        "amount": 211
    }, {
        "month": "Juni",
        "amount": 50
    }, {
        "month": "Juli",
        "amount": 362
    }]
};
```

Uiteindelijk zorgen we ervoor dat deze data gefetcht wordt uit onze MongoDB Database, dit gebeurd onder andere met de onderstaande lijn code. Hier worden de eerste 7 dagen uit de database gehaald

```
  // Hier wordt de staansnelheid data gefetcht uit de database
fetch('/api/staanSnelheid',{
  headers:{
  limit:7,
  }
}).then(result => {
  return result.json();
  }).then(dataSnelheid => {
  console.log(dataSnelheid);
```

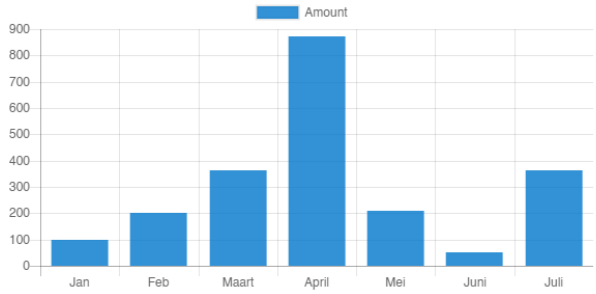
Iteraties Grafieken

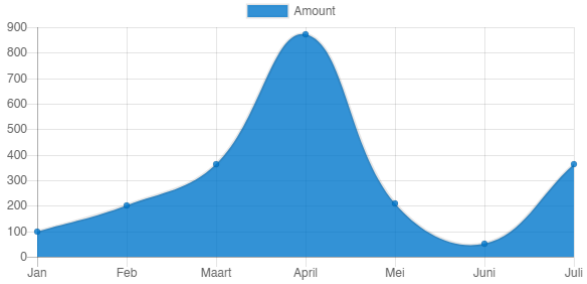
Je kunt het uiterlijk van de grafiek veranderen door in onderstaande code veranderingen toe te passen. Hier kun je bijvoorbeeld de kleur of type grafiek veranderen. Ik ben hier een beetje mee gaan spelen om te kijken wat de beste en duidelijkste manier is om data te laten zien.

```
var ctx = canvas.getContext('2d');  // Hier kun je dingen aanpassen van de grafiek
var config = {
   type: 'bar',        // Je kunt type bijvoorbeeld naar line veranderen voor een lijngrafiek
   data: {
      labels: labels,
      datasets: [{
         label: 'Amount',
         data: data,
         backgroundColor: 'rgba(0, 119, 204, 0.8)' // Hier kun je de kleuren van de grafiek aanpassen
      }]
   }
};

var chart = new Chart(ctx, config);
```

Hieronder komen een aantal grafieken die ik gemaakt heb, deze heb ik uiteindelijk getest bij de eindgebruiker door ze in de app te zetten en vragen te stellen om erachter te komen of ze de data goed konden begrijpen.



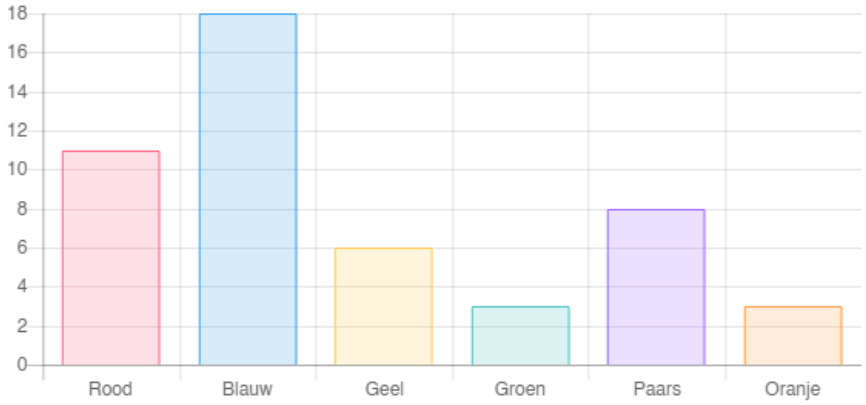


Hierboven zie je de meest basic grafieken. Door de type te veranderen tussen Bar en Line kun je een simpele lijngrafiek en staafdiagram maken.

Feedback

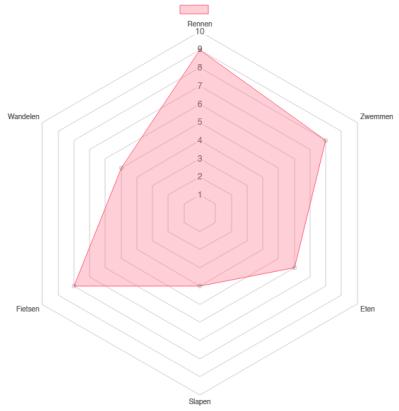
Toen ik deze grafieken aan Ferry liet zien vertelde hij me dat het belangrijk is om met meer kleur te werken. Kleur is een uitstekende manier om verschillende soorten data te onderscheiden, bijvoorbeeld lopen en rennen of andere soorten data. Daarom begon ik te kijken hoe ik verschillende kleuren kon gebruiken in een grafiek.

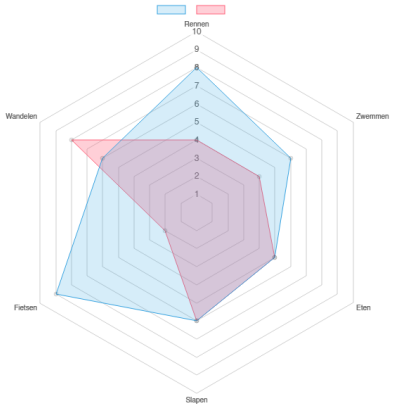
Iteraties



Ook is het mogelijk om elke bar een andere achtergrond kleur te geven en een andere borderkleur. Hierdoor krijg je een mooie staafdiagram zoals hierboven weergegeven. Als je beginAtZero; true gebruikt begint de lijn links altijd op 0, hierdoor verdwijnt de laagste staaf niet. De code voor de grafiek hierboven, zie je hieronder.

```
<script>
var ctx = document.getElementById('canvas').getContext('2d');
var myChart = new Chart(ctx, {
    type: 'bar',
    data: {
        labels: ['Rood', 'Blauw', 'Geel', 'Groen', 'Paars', 'Oranje'],
        datasets: [{
            label: '',
            data: [11, 18, 6, 3, 8, 3],         // Dit is de data die nu gebruikt word, later komt dit uit de database
            backgroundColor: [
                'rgba(255, 98, 122, 0.2)',      // Hier geef je de achtergrondkleuren mee, de 0.2 op het einde geeft de opacity aan
                'rgba(54, 163, 225, 0.2)',
                'rgba(255, 205, 87, 0.2)',
                'rgba(75, 191, 193, 0.2)',
                'rgba(153, 103, 256, 0.2)',
                'rgba(255, 158, 65, 0.2)'
            ],
            borderColor: [
                'rgba(255, 98, 122, 1)',        // Hier bepaal je de kleuren van de lijntjes om een bar heen
                'rgba(54, 163, 225, 1)',
                'rgba(255, 205, 87, 1)',
                'rgba(75, 191, 193, 1)',
                'rgba(153, 103, 256, 1)',
                'rgba(255, 158, 65, 1)'
            ],
            borderWidth: 1
        }]
    },
    options: {
        scales: {
            yAxes: [{
                ticks: {
                    beginAtZero: true // Hiermee begint de grafiek op 0, anders zou de laagste grafiek altijd weg zijn wat een vertekend beeld geeft
                }
```

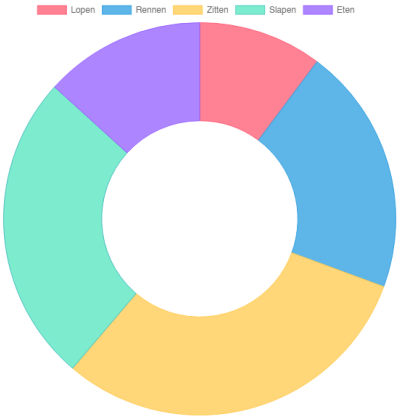




Ook kwam ik erachter dat je een radar grafiek kunt maken, dit werkt bijna hetzelfde alleen moet je meerdere datasets aanmaken om verschillende radargrafieken over elkaar heen te kunnen laten zien. Zoals je hierboven ziet. De code die hierbij hoort zie je hieronder.

Je ziet dat er nu meerdere datasets zijn, de type naar radar is verandert en dat je bij ticks een minimale en maximale waarde moet invullen voor de middelste getallen in de grafiek.

```
var ctx = document.getElementById('canvas').getContext('2d');
var myChart = new Chart(ctx, {
    type: 'radar',
    data: {
        labels: ['Rennen', 'Zwemmen', 'Eten', 'Slapen', 'Fietsen', 'Wandelen'],
        datasets: [{
            label: '',
            data: [8, 6, 5, 6, 9, 6],         // Dit is de data die nu gebruikt word, later komt dit uit de database
            backgroundColor: [
            'rgba(54, 163, 225, 0.2)',     // Hier geef je de achtergrondkleuren mee, de 0.2 op het einde geeft de opacity aan
            ],
            borderColor: [
            'rgba(54, 163, 225, 1)',       // Hier bepaal je de kleuren van de lijntjes om een bar heen
            ],
            borderWidth: 1
        },
        {
            label: '',
            data: [4, 4, 5, 6, 2, 8],         // Dit is de data die nu gebruikt word, later komt dit uit de database
            backgroundColor: [
                'rgba(255, 98, 122, 0.3)',     // Hier geef je de achtergrondkleuren mee, de 0.2 op het einde geeft de opacity aan
            ],
            borderColor: [
                'rgba(255, 98, 122, 1)',       // Hier bepaal je de kleuren van de lijntjes om een bar heen
            ],
            borderWidth: 1
        }]
    },
    options: {
    scale: {
        angleLines: {
            display: false
        },
        ticks: {
            suggestedMin: 0,       // Hiermee zet je de minimale waarde die getoond word in de radargrafiek, als je dit niet doet zal het laagste punt in het midden staan
            suggestedMax: 10    // Hiermee zet je de maximale waarde, als deze te hoog is word het oppervlakte erg klein
        }
    }
}
```



Toen ik verder in de documentatie van Chart.js keek zag ik dat je ook een taart en donutdiagram kunt maken. Deze zie je hierboven. Hiervoor moet je de type veranderen naar ‘pie’ of ‘doughnut’. Het lukte me niet om de grafiek werkend te krijgen zonder het options gedeelte weg te halen. Hierdoor kon ik niet echt extra options toepassen zoals in de documentatie van Chart.js beschreven stond.

```
var ctx = document.getElementById('canvas').getContext('2d');
var myChart = new Chart(ctx, {
    type: 'doughnut', //Verander dit naar 'pie' om een taartdiagram zonder gat in het midden te krijgen
    data: data = {
    datasets: [{
        data: [10, 20, 30, 25, 13],
        backgroundColor: [
                'rgba(255, 98, 122, 0.8)',     // Hier geef je de achtergrondkleuren mee, de 0.8 op het einde geeft de opacity aan
                'rgba(54, 163, 225, 0.8)',
                'rgba(255, 205, 87, 0.8)',
                'rgba(93, 231, 193, 0.8)',
                'rgba(153, 103, 256, 0.8)'
            ],
            borderColor: [
                'rgba(255, 98, 122, 1)',       // Hier bepaal je de kleuren van de lijntjes om een gedeelte heen
                'rgba(54, 163, 225, 1)',
                'rgba(255, 205, 87, 1)',
                'rgba(75, 191, 193, 1)',
                'rgba(153, 103, 256, 1)'
            ],
            borderWidth: 1
    },
    ],

    // Deze labels komen bovenin de legenda te staan
    labels: ['Lopen','Rennen','Zitten','Slapen','Eten']
},
//     options: { er worden geen extra options gebruikt dus dit kan weg
```

De documentatie van Chart.js kun je vinden op https://www.chartjs.org/docs/latest/
Hier heb ik enorm veel gebruik van gemaakt en hierdoor leerde ik snel hoe deze library in elkaar zit.

De test poc’s van alle charts kun je vinden in mijn gitlab repository op
https://git.fhict.nl/I437759/semester-3-media/-/tree/master/Poc's/Chart.js%20poc

Feedback/Gebruikerstest

Voordat ik bij elke grafiek alle opties zou verkennen wou ik eerst weten welke grafieken het beste waren om te gebruiken. Ik zette al deze grafieken in de app zodat ik deze via de telefoon kon testen bij de eindgebruikers. In deze test stelde ik de vraag wat de waarde van een bepaald gegeven was. Zo vroeg ik welke waarde er bij lopen hoorde in de lijn grafiek, staafdiagram, taartdiagram etc.

Hieruit bleek dat je een radargrafiek het meest ingewikkelde was, dit vonden de ouderen niet overzichtelijk en ze wisten niet goed welke waarde je moest aflezen. De taartdiagram was duidelijker maar ze merkte niet altijd dat je op een gedeelte kunt klikken om de waarde te zien, het is lastig om een exacte waarde te gokken in een taartdiagram. Ook kun je niet echt snelheid weergeven in een taartdiagram.

Uiteindelijk bleek dat de simpele staafdiagram het beste was omdat je hier alle data overzichtelijk gesplitst hebt én je de waardes links af kunt lezen. Je hoeft dus niet op een staaf te klikken voordat je de waardes ziet. Hetzelfde geld voor een lijngrafiek, alleen is de lijngrafiek duidelijker om snelheden weer te geven.

Toen ik dit aan Ferry liet zien was hij het ermee eens. Hij vond het wel handig als ik even zou kijken naar de grootte van de cijfers. Deze waren nog erg klein. Ik moest dus even kijken of ik font en grootte aan kon passen zodat dit bij ons design paste.

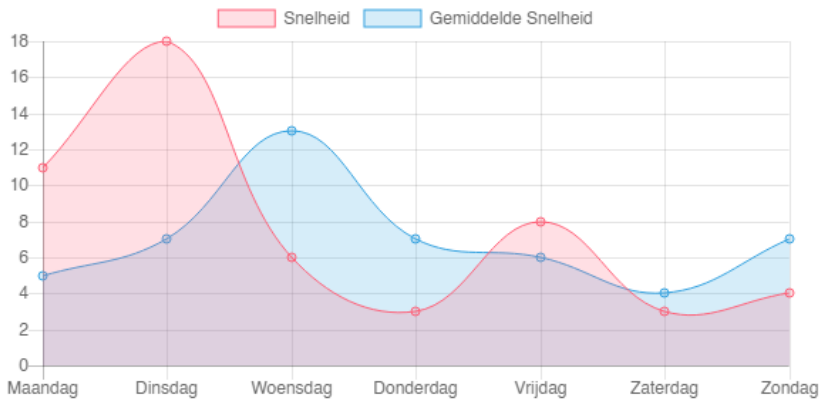
Ook vertelde Olaf dat ik goed moet nadenken over het kleurgebruik. Blauw is een goede neutrale kleur om je data mee aan te geven maar als je rood of groen gebruikt kan dit opgevat worden als goed of slecht. Dit nam ik mee in mijn gebruikersonderzoeken en het bleek inderdaad dat mensen snel schikken van de kleur rood, hierdoor denken ze dat er iets erg fout is. Daarom heb ik uiteindelijk besloten om de waarschuwingskleur oranje te maken in plaats van rood.

Conclusie

Blauw is een goede kleur omdat het neutraal is, dit geeft geen slecht of goed gevoel. Als je een rode kleur gebruikt schrikken mensen vaak meteen en denken ze dat er iets ernstigs aan de hand is, daarom heb ik gekozen voor een oranje kleur als waarschuwingskleur.

Uit de gebruikerstesten bleek dat de radargrafiek en de taartdiagram niet de beste keus was. Uiteindelijk bleek dat je voor aantallen het beste de staafdiagram kunt gebruiken en voor snelheden het beste de lijndiagram kan gebruiken.

Toen ik de andere grafieken aan het maken was leerde ik steeds meer over Chart.js, dit is ook omdat ik de documentatie er goed bijhield. Eerst snapte ik niet hoe je meerdere lijnen of stukken in dezelfde grafiek kunt stoppen, uiteindelijk moest ik dit doen bij de radargrafiek. Hierdoor lukte het me uiteindelijk om dit ook bij een lijngrafiek te doen. Hierdoor kan ik de snelheid en gemiddelde snelheid overzichtelijk weergeven in dezelfde grafiek zoals hieronder is te zien, dit is precies wat we nodig hebben.



Reflectie

Eerst kon ik nog niet goed overweg met een library, door de documentatie goed te lezen heb ik veel geleerd over het werken met Chart.js en kan ik alle grafieken mooi aanpassen.

Ik heb geleerd hoe ik goede gebruikerstesten uitvoer zonder direct te vragen wat iemand fijner vind. Het is belangrijk om hierachter te komen door open vragen te stellen of door mensen iets te laten doen zonder zelf teveel informatie te geven. Ik heb vervolgens met deze resultaten én feedback van de docenten nieuwe iteraties gemaakt en verder gewerkt.

Voor de volgende keer zou ik graag meer testen willen doen, door corona kon ik nu niet veel gebruikerstesten doen maar hiervoor heb ik extra feedback van de docenten gevraagd. Hierdoor kreeg ik uiteindelijk toch nog genoeg feedback om nieuwe iteraties te maken.