l

Tobias Gaarenstroom

Havo 4

Songtekst Processor

Natuurkunde Onderzoek

Inhoudsopgave

[Onderwerp/inleiding 1](#_Toc168651694)

[Aanleiding voor het onderzoek 1](#_Toc168651695)

[Hoofdvraag 2](#_Toc168651696)

[Deelvragen 2](#_Toc168651697)

[Waar kun je songteksten vandaan halen? 2](#_Toc168651698)

[Hypothese 2](#_Toc168651699)

[Uitwerking 2](#_Toc168651700)

[Wat is er allemaal nodig om songteksten goed te kunnen formateren? 3](#_Toc168651701)

[Hypothese 3](#_Toc168651702)

[Uitwerking 3](#_Toc168651703)

[Welke techniek is het beste om een processor te maken? 4](#_Toc168651704)

[Hypothese 4](#_Toc168651705)

[Uitwerking 4](#_Toc168651706)

[Conclusion 5](#_Toc168651707)

[Verwijzingen/Bronnen 7](#_Toc168651708)

[Extra dingen 7](#_Toc168651709)

[Planning 7](#_Toc168651710)

[Logboek 7](#_Toc168651711)

# Onderwerp/inleiding

**Songtekst processor**

De songtekst processor gaat een script worden die songteksten kan ophalen, doorlezen en in logische blokken opdelen/samenvoegen. En deze dan later (zonder te veel gedoe en code) ook nog te kunnen tonen in een front-end, zoals met HTML en CSS.

# Aanleiding voor het onderzoek

Ik heb dit gekozen omdat het heel erg handig is om een code te hebben die songteksten kan uitlezen en zo vormen dat het bij alles werkt. En zo kan ik er ook meteen van kan profiteren, want als je een songtekst-projectie-software maakt, is het wel handig om een manier te hebben om songteksten te kunnen lezen en weergeven op een manier waarop het logisch is opgebouwd en dat het handig is om mee te kunnen werken.

# Hoofdvraag

**Wat is er nodig om songteksten goed, efficiënt en handig te kunnen processen.**

Hierbij bedoelen we dat we van songtekst met ‘placeholders’ en andere markeringen een lijst met stukjes van de songtekst maken. Denk bij ‘placeholders’ aan dingen waarmee je makkelijk herhalingen, vertalingen en soort gelijke dingen mogelijk te maken.

# Deelvragen

Om de hoofdvraag te kunnen beantwoorden, moeten we de vraag in meerdere kleinere vragen opdelen. Deze kleinere vragen staan hier beneden met kop 2 aangeduid. Daarin zit dan in kop 3 de hypothese en de uitwerking ervan.

## Waar kun je songteksten vandaan halen?

### Hypothese

Ik denk dat we songteksten in elk geval van het internet kunnen krijgen, maar of er ergens een plek is waar je ze allemaal vandaan kunt halen is de vraag.

\*Als deze deelvraag niet (goed) te beantwoorden valt achteraf, dan pak ik gewoon zelf wat songteksten om daarmee verder te kunnen werken\*

### Spotify - Muziek en podcasts - Apps op Google PlayUitwerking

Voordat ik begon met zoeken had ik al wel een idee om bij Spotify te kijken, want die streaming dienst geeft je de songtekst van je huidige lied, zodat je die meekunt zingen. Maar zoals wel logisch, kun je niets vinden in Spotify zelf. Toen ik ging zoeken op google kwam ik langs een artikel (Jenny, 2021) waarin ze zeggen dat Spotify gebruik maakt van Musixmatch en Genius. Dus daar ging ik kijken.

Bij Musixmatch kon ik eerst vrij weinig vinden, maar toen kwam ik de “For developers” link tegen en ging kijken. Voor hun API heb je een API-sleutel nodig. In principe is dit voor privé geen probleem, maar voor publiek gebruik is dat onhandig. Vooral omdat je een rate limit hebt en vanwege dat het lastig is om de API-sleutel te kunnen verbergen in je code.

Afbeelding met clipart, Graphics

Automatisch gegenereerde beschrijvingBij Genius is het eigenlijk hetzelfde verhaal. Maar hier moet je met je eigen account inloggen voor een access token, dus dat wordt al helemaal niks.

Toen bedacht ik me dat Google ook songteksten levert. Ik ben wat songteksten wezen testen en die werkten. Daarbij stond ook waar de tekst vandaan komt en dat leek via verschillende API’s te gaan.

Afbeelding met Graphics, logo, symbool, cirkel

Automatisch gegenereerde beschrijvingAls ik op een manier het zoek resultaat van google kan ophalen en de songtekst kan verkrijgen, dan hoef ik niet moeilijk te doen om API-sleutels te hoeven verbergen en zo. Alleen weet ik niet of dit legaal is, maar het iets wat ik kan proberen.

## Wat is er allemaal nodig om songteksten goed te kunnen formateren?

### Hypothese

Ik denk dat we zo ie zo wel een aanduiding voor refrein en mogelijk andere dingen als bridge en zo nodig hebben. En daarmee dan ook de tekst niet te hoeven herhalen, maar gewoon de naam kunnen refereerden en zo de tekst in te vullen. En ook is het handig/nodig om stukken te kunnen herhalen.

### Uitwerking

Om alles zo mooi mogelijk te maken kunnen we de volgende dingen gebruiken, lijkt mij:

Refrein:  
dit is mijn refrein!

(Refrein 2x)

Deze zin moet je drie keer zingen (3x)

Deze alinea  
of hoe je het wilt noemen  
zou toch eens twee keer moeten  
Dus daarom deze  
(2x)

Repeat Refrein

In dit blok  
moet een split  
[split]  
zodat het nog steeds hetzelfde ding is  
maar op meerdere blokjes  
[split]  
zodat we dit bijv. als een refrein kunnen noteren  
zonder dat het lastig word

Maar het eindigen op een ‘:’ kan ook in de songtekst zelf zitten, maar de aanduiding zou niet meer dan 1 woord moeten zijn, dus dan is dat opgelost.

Maar ja dat is niet alles, want we kunnen ook nog dingen aanduiden in de eerste aantal regels misschien, door te beginnen met ‘#’

# TITLE=Dit is mijn titel; WITH\_TRANSLATION=1

This is my English text  
Dit is mijn Engelse tekst.

Dus dat zijn in elk geval dingen die we kunnen gebruiken/toevoegen. Want niet al deze dingen worden standaard gebruikt, maar ‘Refrein:’ en ‘Refrein 2x’ komt wel eens voor in songteksten, zoals in [“De kabouterdans”](https://www.songteksten.nl/songteksten/19545/kabouter-plop/de-kabouterdans.htm) (Plop). Daar wordt gebruik gemaakt van ‘Chorus:’ en ‘Repeat Chorus’.

## Welke techniek is het beste om een processor te maken?

### Hypothese

Ik denk dat we misschien iets als een regex (of reguliere expressie) kunnen gebruiken of een per lijn tekst processor.

“Een reguliere expressie is een is een manier om patronen te beschrijven waardoor een computer softwarematig tekst kan herkennen.” (Reguliere expressie, 2020)

### Uitwerking

Allereerst slaan we een songtekst op, bijvoorbeeld de kabouterdans die eerder benoemd is.

We moeten zo ie zo eerste alle alinea’s scheiden, aangezien daar dus altijd 2 enters tussen zitten, zouden we kunnen splitten op “\n\n”. Dit kan worden gedaan met code, bijvoorbeeld de volgende javascript code:

|  |
| --- |
| const str = 'MIJN\_SONG\_TEKST'; // defineer de songtekst  let paragraphs = str.split('\n\n'); // split de alinea's |

Daarnaast moeten we dus de ‘\n[split]\n’ dingen splitten, zonder dat het echt een nieuwe alinea is, dus eigenlijk moeten voor elke alinea eerst hem in een enkele array zetten, [[‘alinea1’], [‘alinea 2’]] en daarna op het ‘\n[split]\n’ splitten

|  |
| --- |
| paragraphs = paragraphs.map((al) => [al]); // map de paragraphs in losse arrays  paragraphs = paragraphs.map((al) => al[0].split('\n[split]\n')); // split op het '\n[split]\n' woordje |

Nu moeten we elke alinea langs om te checken of de eerste lijn eindigt op een ‘:’ en maar 1 woord is, dus zonder spaties (‘ ’).

|  |
| --- |
| let parts = paragraphs.filter((al) => al[0].split('\n')[0].endsWith(':') && al[0].split('\n')[0].split(' ').length == 1); // filter alle die eindigen op ':' en geen spaties bevatten  let blockNames = parts.map((al) => al[0].split('\n')[0].replace(':', '')); // verkrijg de namen aanduiding uit de parts array  let blockContents = parts.map((al) => { const lines = al[0].split('\n'); lines.splice(0,1); al[0] = lines.join('\n'); return al; }); // verkrijg de inhoud, doormiddel van het verwijderen van de eerste uit de parts array  let blocks = blockNames.map((name, i) => [name, blockContents[i]]); // combineer de twee lijsten tot 1 lijst met ['name', 'content'] items |

Nu dat we een lijst met de blok namen hebben en de inhoud ervan in één lijst kunnen verder zoeken naar waar de blokken gebruikt worden. Hiervoor checken we of de alinea 1 lijn is en of blockNames de lijn bevat (zonder Repeat en haakjes dingen)

|  |
| --- |
| let blockUsage = paragraphs.map((al, i) => { const p = al[0].split('\n'); return (al.length > 1 || p.length > 1) ? false : (blockNames.includes(p[0]) || blockNames.includes(p[0].replace('Repeat ', '')) || blockNames.includes(p[0].replace(/\(([\w]+)( (\d+)x|)\)/, '$1'))) ? [al, i] : false;}).filter((al) => al[0]); // filter op alinea's waarbij de inhoud in de blockNames lijst zit en behoud de originele index  blockUsage = blockUsage.map((us) => [us[0][0].replace('Repeat ', '').replace(/\((\w+)( (\d+)x|)\)/, '$1$2')]); // Verwijder 'Repeat ' van tekst en de '(key)' en '(key 2x)' |

Effe in een [naam, aantal, tekst] formaat

|  |
| --- |
| let blockUsing = blockUsage.map((us)=>{  const ourid = us[0].replace(/ (\d+)x/, '');  const amount = Number(us[0].replace(/(\w+)( (\d+)x|)/, '$3')) || 1;  out = blocks.find(([id,v]) => id == ourid)  return [ourid, amount, out[1]]; // [blockName, aantal, tekst]  }); |

En dan hebben we genoeg informatie die we dan kunnen gebruiken om de songtekst compleet te maken.

Then we can use this part to actually fill in the paragraphs

|  |
| --- |
| this.coupletsWithReferences = JSON.parse(JSON.stringify(paragraphs));  for (const [index, blockName, amount, text] of blockUsing) {  paragraphs[index] = text; // replace the contents of the paragraph with the contents of the reference  for (let i=1;i<amount;i++) paragraphs[index] = [...paragraphs[index], ...text]; // repeat adding in the contents for the amount  }  this.couplets = paragraphs; |

# Conclusion

We zijn erachter gekomen dat google de beste plek is om songteksten vandaan te kunnen halen, wat er allemaal nodig is om songteksten goed te kunnen formateren en welke technieken het beste zijn om songteksten te kunnen processen/parsen.

En dit is alle code die je nodig hebt om songtekst in een array te krijgen (NOG NIET helemaal…):

|  |
| --- |
| const str = 'Refrein:\nBlah Die Blah\nDie Blah Blah\n\nWowie wowie\nDusie busie\n[split]\nYeah Yeah\nOh Oh\n\nRepeat Refrein\n\n(Refrein)'; // defineer de songtekst  let paragraphs = str.split('\n\n'); // split de alinea's  paragraphs = paragraphs.map((al) => [al]); // map de paragraphs in losse arrays  paragraphs = paragraphs.map((al) => al[0].split('[split]')); // split op het '\n[split]\n' woordje  const lastPart = paragraphs[paragraphs.length - 1],lastAl = lastPart[lastPart.length - 1]; // get the last array of the last paragraph  if (lastAl.endsWith('\n')) paragraphs[paragraphs.length - 1][lastPart.length - 1] = lastAl.slice(0, -1); // Make sure to remove the linebreak of the last paragraph, else some statements will fail  let parts = paragraphs.filter((al) => al[0].split('\n')[0].endsWith(':') && al[0].split('\n')[0].split(' ').length == 1); // filter alle die eindigen op ':' en geen spaties bevatten  let blockNames = parts.map((al) => al[0].split('\n')[0].replace(':', '')); // verkrijg de namen aanduiding uit de parts array  let blockContents = parts.map((al) => { const lines = al[0].split('\n'); lines.splice(0,1); al[0] = lines.join('\n'); return al; }); // verkrijg de inhoud, doormiddel van het verwijderen van de eerste uit de parts array  let blocks = blockNames.map((name, i) => [name, blockContents[i]]); // combineer de twee lijsten tot 1 lijst met ['name', 'content'] items  let blockUsage = paragraphs.filter((al) => { const p = al[0].split('\n'); return (al.length > 1 || p.length > 1) ? false : blockNames.includes(p[0]) || blockNames.includes(p[0].replace('Repeat ', '')) || blockNames.includes(p[0].replace(/\(([\w]+)( (\d+)x|)\)/, '$1'));}); // filter op alinea's waarbij de inhoud in de blockNames lijst zit  blockUsage = blockUsage.map((us, i) => [ us[0].replace('Repeat ', '').replace(/\((\w+)( (\d+)x|)\)/, '$1$2'), i]); // Verwijder 'Repeat ' van tekst en de '(key)' en '(key 2x)'  let blockUsing = blockUsage.map((us)=>{  const ourid = us[0].replace(/ (\d+)x/, '');  const amount = Number(us[0].replace(/(\w+)( (\d+)x|)/, '$3')) || 1;  out = blocks.find(([id,v]) => id == ourid)  return [us[1], ourid, amount, out[1]]; // [index in paragraphs, blockName, aantal, tekst]  });  this.coupletsWithReferences = JSON.parse(JSON.stringify(paragraphs));  for (const [index, blockName, amount, text] of blockUsing) {  paragraphs[index] = text; // replace the contents of the paragraph with the contents of the reference  for (let i=1;i<amount;i++) paragraphs[index] = [...paragraphs[index], ...text]; // repeat adding in the contents for the amount  }  this.couplets = paragraphs; |

Dus dat is alles wat nodig is!

# Verwijzingen/Bronnen

Jenny, L. (2021, 11 24). *​Waar haalt Spotify eigenlijk die songteksten vandaan?* Opgehaald van DutchCowboys: https://www.dutchcowboys.nl/entertainment/waar-haalt-spotify-eigenlijk-die-songteksten-vandaan

Plop, K. (sd). *De Kabouterdans*. Opgeroepen op Mei 23, 2023, van Songteksten.nl: https://www.songteksten.nl/songteksten/19545/kabouter-plop/de-kabouterdans.htm

*Reguliere expressie*. (2020, januari 2). Opgehaald van Wikipedia: https://nl.wikipedia.org/wiki/Reguliere\_expressie

De voorbeeld (javascript) code is in kleurtjes dankzij <http://hilite.me/> (Met stijl ‘vs’)

# Extra dingen

## Planning

|  |  |
| --- | --- |
| Week 19-20 (8-5) | Onderzoeken wat er mogelijk is voor de eerste vraag |
| Week 21-22 (22-5) | Eerste + tweede deelvraag beantwoorden/concluderen |
| Week 23 (8-6) | Derde deelvraag beantwoorden/concluderen |
| Week 24 (15-6) | Conclusie trekken |

## Logboek

10-5-2023: Vooral rondgekeken op internet (1 lesuur)  
12-5-2023: Informatie vergaard voor deelvraag 1 (1 of 2 lesuren)

17-5-2023: Meer info en conclusies (1 lesuur)

25-5-2023: Gewerkt aan deelvraag 2 (1 lesuur)  
26-5-2023: 3 lesuren gewerkt aan het verslag + deelvraag 2

8-6-2023: Gewerkt aan deelvraag 3 (1 lesuur)  
15-6-2023: Gewerkt aan deelvraag 3 + conclusie voorbereiden (1 lesuur)   
18-6-2023: Gewerkt aan het compleet maken van onderzoek (1-2 uurtjes)

7-6-2024: Updated the code and translated a little tiny bit for the release of plaintext-song-parser.js