Vue.js vs Angular

**Introductie:**

In deze paper vergelijk ik 2 bekende en veel gebruikte Javascript bibliotheken. In een korte introductie, geef ik wat meer informatie over de frameworks. Wanneer en waarom zijn ze ontstaan en wie zijn de ontwikkelaars.

AngularJS:

Angular is geboren als GetAngular, in 2009 opgericht door een aantal ingenieurs waaronder Misko Hevery. Het werd redelijk snel populair nadat Misko een webapplicatie opnieuw ontwikkelde met GetAngular en de oorspronkelijke hoeveelheid code deelde door 17.

Google merkte dit op en begon meteen met sponsoring. Google maakte er een open-source framework van: AngularJS. Met de belangrijkste kenmerken: ‘Two-way-data binding’, ‘Dependency Injection’ en het veelvuldig gebruik van HTML-attributen werd het al snel een populair framework.

Vue.js:

Vue is een veel jonger en kleiner framework. Het is opgericht in 2014 door Evan You en het is gebaseerd op AngularJS en Backbone. Hoewel het zich nog in de beginperiode bevindt, is het al razend populair en veel kenners zeggen dat de populariteit doorheen de jaren enkel nog gaat stijgen. Veel websites zoals ‘Javascript Weekly’ en ‘HackerNews’ prijzen het ook als een zeer beloftevol framework. De nieuwste versie is Vue 2.0.

**Gelijkenissen**:

Vue en Angular lijken hard op elkaar. Dit ligt aan het feit dat Evan You (de ontwikkelaar van Vue) zich mede heeft gebaseerd op AngularJS bij het ontwikkelen van Vue. Het zijn beide open-source frameworks dus iedereen kan ze gebruiken en er zelf aanpassingen aan uitvoeren (voor eigen gebruik).

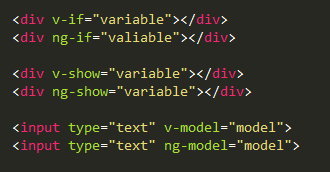
Eerst en vooral richten de 2 frameworks zich op dezelfde soort van applicaties. De meeste dingen die je met Vue kan, kan je ook met Angular. Ze worden namelijk allebei het meest gebruikt om een ‘one-page-website’ te maken. Een website die zo weinig mogelijk data communiceert met de server. In plaats van per ‘POST’ of ‘GET’ een hele pagina op te vragen aan de server, kan je met Javascript een ‘Json request’ doen en zo enkel de data opvragen die je nodig hebt. Zo zal je pagina niet telkens helemaal moeten herladen en zal alles sneller en gebruiksvriendelijker werken.

Ze proberen beide ook het mv\* principe te gebruiken. ‘Model-View-Wildcart’ (mvc, mvvm,…). Je kan het dus gebruiken om je code te organiseren op een universele manier. Je bent in die structuur ook enorm vrij bij beide frameworks.

De 2 frameworks werken ook goed samen met Laravel. Ik heb zelf 2 zeer kleine applicaties gemaakt met de 2 frameworks in hetzelfde Laravel- project en de samenwerking loop zeer vlot.

Met Vue kan je zelfs je initiële data rechtstreeks binnenkrijgen via Laravel , zonder eerst een “Ajax -call” te moeten doen, dit doe je met behulp van “props” en “parent-child” communicatie.

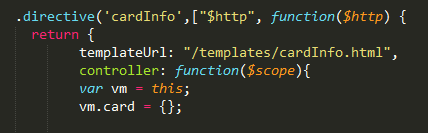
Wat de 2 typeert, zijn vooral de HTML attributen. Het zijn 2 frameworks die veel gebruik maken van HTML attributen om zo een duidelijke communicatie te maken tussen de Javascript en de HTML. Zo heb je “ng-model” voor Angular en “v-model” voor Vue, beiden hebben dezelfde functionaliteit. Er zijn nog andere HTML attributen die hard op elkaar lijken.

Hier zie je 3 veel gebruikte HTML attributen voor beide frameworks. Alles wat met een ‘v’ begint is van Vue, alles was met ‘ng’ begint is van Angular.

Het gebruik van al deze attributen is zeer kenmerkend voor deze frameworks. Ook het werken met de dubbele accolades is een typisch kenmerk.

Om je code te structureren, gebruiken beide frameworks een manier om code te scheiden, op te delen in overzichtelijke delen. Zo gebruikt Angular Directives. Dit is een apart stuk Javascript en HTML die je onafhankelijk van elkaar kan laten werken. Bij Angular kan je het stuk HTML dat er bij hoort ook in een andere file opslaan. Dit zorgt voor een veel overzichtelijkere code.

Ook Vue hanteert een zelfde soort principe maar noemt dit Componenten. Componenten zijn kleine afgescheiden stukken HTML en Javascript die je meerdere keren en onafhankelijk van elkaar kan gebruiken.



Hier zie je een directive in Angular. Achter de variabele “templateUrl” zie je de path waar het stuk HTML zit dat hoort bij deze directive. Je ziet ook wat variabelen zoals “vm.card”. De scope van deze variabelen bestaat enkel binnen deze directive.



Hier zie je een component in Vue. Achter de variabele template zie je een ‘id’ waar het stuk HTML van deze component zit. De id wordt gebruikt in de HTML pagina. Hier worden de stukken HTML niet in aparte kleine files gestopt. Dit kan wel gedaan worden met het installeren van een ‘package’.

In de data functie zie je ook een paar variabelen die een scope hebben, enkel binnen dat component.

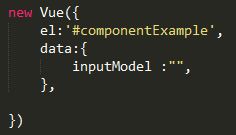
Zoals je ziet, zijn er wel wat gelijkenissen tussen componenten bij Vue en de directives bij Angular.

Bij zowel Vue als Angular kan je ook gebruik maken van filters. Dit is een heel gemakkelijke en snelle manier om data te vervormen naar een bepaald type. Veel filters doen hetzelfde maar ze krijgen meestal wel een andere naam. Ze worden bij beide frameworks onderscheiden van de rest met een “|” teken.

Links zie je een filter in werking bij Vue, rechts een filter bij Angular. Ze geven beide hetzelfde resultaat.

Two-way data binding is ook 1 van de belangrijkste kenmerken van zowel Vue als Angular. Dit is een zeer praktisch en veel gebruikt principe omdat er veel “achter de schermen” gebeurt, waarvan de ontwikkelaar zich niets moet aantrekken. Bij beide frameworks werkt dit met het attribuut “model”.

Two-way data binding houdt in dat data wordt geüpdatet in 2 richtingen. Als je een bepaalde variabele vasthangt aan een input met “ng-model” bij Angular of “v-model” bij Vue, zal een wijziging van het de input variabele of een wijziging van de variabele in de Javascript zorgen voor een algemene verandering van de variabele. Zo kan je data doorgeven naar de Javascript en hierop reageren zonder een “event trigger” te gebruiken (bijvoorbeeld een key-down). Met “computed properties “of “$watch” events wordt dit nog interessanter maar hier kom ik later nog op terug.

Met two-way binding is het mogelijk om iets in een input te typen en het real time te printen met enkel een paar lijnen code.

Met deze paar lijnen code wordt het” h1 element” per “key-down” aangepast. Dit allemaal door een variabele vast te hangen aan een input field.

Al deze gebruiksvriendelijke en intuïtieve zaken zorgen voor 2 zeer praktische frameworks die beide aan te raden zijn bij het maken van een single web application.

Nu we de gelijkenissen tussen Angular en Vue kennen, bespreken we de verschillen.

**Verschillen:**

Ik heb 2 zeer simpele single web applicaties ontwikkeld met behulp van Laravel als backend framework. De bedoeling van de pagina is dat je een aantal artikels hebt waarvan enkel de titel wordt getoond. Bij het klikken op een artikel wordt het hele artikel getoond + een formulier waarmee je het artikel eventueel kan aanpassen.

Ik heb deze eenvoudige applicatie 2 keer gemaakt. In Vue en in Angular om zo de verschillen tussen de 2 frameworks goed te begrijpen.

Ik ben er zelf nog niet uit welk framework ik zou verkiezen, aangezien ze allebei hun eigen voor- en nadelen hebben.

Eerst en vooral heb ik gemerkt dat ik met Vue een pak minder code heb geschreven.

Links zie je de Javascript code die ik met Angular heb geschreven. Rechts zie je de code die ik met Vue heb geschreven.

Om het verschil duidelijk te maken, heb ik een foto genomen van het uitgezoomde beeld rechtsboven bij “Sublime Text Editor’.

Dit verschil is niet echt een nadeel van Angular, omdat ik vind dat Angular beter gestructureerd is. Bij Angular heb ik 3 directives aangemaakt. Dit was niet echt noodzakelijk voor deze kleine applicatie, maar dit zorgt er wel voor dat een eventuele uitbreiding later gemakkelijk implementeerbaar zal zijn.

De eerste directive zorgt voor het tonen van de lijst met titels. De 2de directive zorgt voor het tonen van het geselecteerde artikel. De 3de directive zorgt voor het kleine formulier dat gebruikt kan worden om een artikel aan te passen.

Structuur:

Het goede aan de applicatie met Angular is ook dat de 3 HTML templates die horen bij deze directives in aparte HTML files konden worden gezet. Dit zorgt voor een enorme boost in de overzichtelijkheid van het project en de herbruikbaarheid van deze directives. Dus de extra lijnen code offer ik graag op aan een gemakkelijkere uitbreidbare en overzichtelijke structuur.

Bij Vue ligt de hoeveelheid code een stuk lager. Dit komt ook omdat ik maar met 1 component heb gewerkt. De reden hiervoor is dat de communicatie tussen verschillende componenten niet heel handig is als je het project kleinschalig wil houden. Bij Vue zou ik een aparte Vue instantie moeten aanmaken en deze gebruiken als “brug” om te communiceren tussen verschillende componenten. Dit vind ik een bizarre manier van werken. Daarom heb ik alles binnen 1 component gebracht. Deze code zal minder goed uit te breiden zijn, maar het behoudt de overzichtelijkheid wel.

Bij Vue heb ik de aparte HTML templates ook niet in aparte files kunnen zetten. Dit is niet bij Vue ingebouwd. In plaats daarvan refereer ik naar een “div” in de HTML pagina waar ik de template heb tussen gezet met behulp van een id-attribuut. Ik moet hierbij opmerken dat het mogelijk is om de templates in aparte files op te slaan, maar dit met behulp van een package.

Vue heeft dit intentioneel zo gedaan en zegt dat het gemakkelijker is voor de ontwikkelaars als de templates in dezelfde file als de componenten staan. Dit omdat ze zo nauw samenwerken en het lastig kan zijn om tijdens het ontwikkelen voortdurend te switchen van bestand.

De ontwikkelaars van Vue beweren ook dat het sneller zal werken, eenmaal het op een server staat. Angular laadt de files in met behulp van “Ajax” en dit zal zorgen voor langere laadtijden.

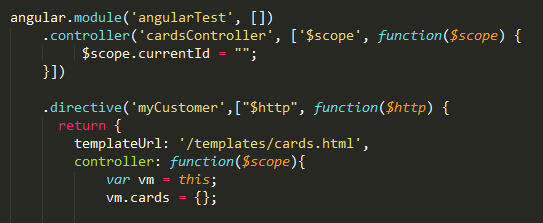
Ik ben het hier niet mee eens. Ik heb nog geen ervaring met grootschalige applicaties in Vue maar ik denk dat het na verloop van tijd chaotisch zal worden als je de verschillende templates niet kan opsplitsen.

Bij Vue vind ik de structuur van de componenten in de Javascript wel overzichtelijk. Je bent gedwongen om alles onder te verdelen in categorieën in de vorm van objecten. Zo heb je een object “methods” waarin je al de methodes moet zetten. Je hebt een object “data” waarin al de variabelen staan en zo gaat het door. Deze structuur is meer overzichtelijk ten opzichte van Angular. Bij Angular kan je natuurlijk zelf een object “methods” aanmaken en hier al de methodes inzetten, maar je bent niet verplicht om dit te doen. Je zal ook een andere manier moeten vinden om te refereren naar gebruikte variabelen. Als je een methode gebruikt in Vue, dan zal Vue ook meteen kijken in het object “methods”: Je moet niet specifiëren in welk object je methode zich bevindt. Dit maakt je Javascript ook heel leesbaar voor iedereen die het niet geschreven heeft.

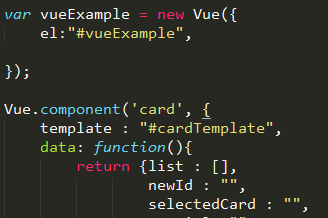
Complexiteit

Tijdens het maken van deze eenvoudige applicatie heb ik gemerkt dat er wel een verschil bestaat in complexiteit. Angular is veel lastiger en frustrerender om mee te werken op vlak van coderen.

Je moet per module ook een “controller” aanmaken die je dan moet gebruiken met het attribuut “ng-controller” in je HTML code. Dit maakt alles wat ingewikkelder en minder duidelijk, zeker voor kleine projecten.



Deze foto toont aan hoe je een nieuwe directive kan aanmaken in Angular. Eerst moet er een module zijn waaraan je een directive koppelt.



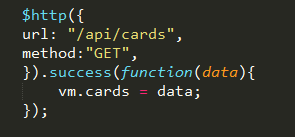
Deze code toont hoe je een component kan aamaken in Vue. Er moet ook een Vue instantie zijn waarin deze gebruikt kan worden. Op deze foto heet deze instantie “VueExample”.

Hieraan zie je duidelijk een verschil in complexiteit. Angular heeft veel meer tools en zorgt er ook voor dat je een complexere applicatie kan bouwen, maar voor kleinschalige projecten is het minder gemakkelijk en zorgt deze grote hoeveelheid tools voor een langere “leercurve” en een ingewikkeldere aanpak.

Ajax:

Om een Ajax call te doen in Angular heeft Angular een functie voorzien. Deze functie is gemakkelijk bruikbaar en past goed in het geheel van de directives en modules. Je moet eerst een “$http object” mee krijgen in je directive en dan kan je die gebruiken om er een Ajax call mee uit te voeren. Als je een one page app wil maken, zal je deze functie ook veel gebruiken. In eender welke applicatie waarmee je aanpassingen wil maken in een databank, zal je dit soort requests moeten maken.

Wat spijtig is bij Vue, is dat ze deze functie niet ingebouwd hebben. Je zal dus een ander framework zoals jQuery moeten inschakelen of het enkel met Javascript schrijven. Als je enkel jQuery zou gebruiken om de Ajax call te doen, is het wat inefficiënt om daarvoor de hele bibliotheek in te laden.

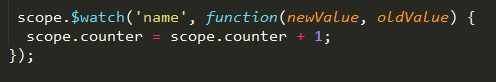
Ik merk wel op dat Vue zeer flexibel is in het samenwerken met andere bibliotheken of frameworks. Er komen geen conflicten en de performance lijdt er niet onder.

Ajax call in Angular. Zoals je ziet, wordt er gebruik gemaakt van “$http”.

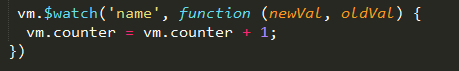
Watchers en computed properties:

Bij zowel Angular als Vue kan je gebruik maken van een “watch-functie”. De functie spreekt voor zich, je houdt een bepaalde variabele in het oog met de “watch-functie”. Als deze variabele gewijzigd wordt, kan je in deze “watch-functie” iets uitvoeren: Zo zou je bijvoorbeeld een paar afhankelijke variabelen mee kunnen laten veranderen.

Dit is een zeer handige functie en als je iets maakt in Vue of Angular zal je hier hoogstwaarschijnlijk gebruik van maken. In beide frameworks wordt de watch functie bijna hetzelfde gebruikt.



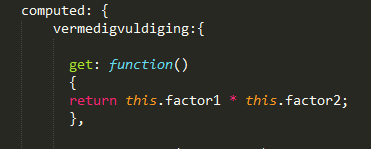
Watch functie in Angular.



Watch functie in Vue.

Zoals je ziet, zijn deze bijna identiek . Bij Angular heeft je “watch” enkel een “$scope-object” nodig. Maar dit is geen hindernis.

Vue heeft een zeer gemakkelijk alternatief voor een” watch- functie”: een ‘computed property’. Dit is zeer gemakkelijk te gebruiken en ik merk vaak dat je hiermee een “ watch- functie” overbodig maakt. Het maakt je code kleiner en gemakkelijker te begrijpen.

Een “ computed property” is een functie die zichzelf herlaadt als een afhankelijke variabele verandert.

Bij de “computed property” die je op de foto ziet, zal de vermenigvuldiging zichtzelf herladen als 1 van de 2 factoren verandert.

Je zal dus geen “ watch- functie” moeten aanzetten om de 2 factoren in het oog te houden om daarna , bij een wijziging, een functie uit te voeren. De “computed property” doet dit allemaal voor jou.

Dit vind ik persoonlijk een enorm voordeel bij Vue.js. Problemen met asynchrone zaken zoals een “Ajax call” lost dit volledig op. Je moet niet wachten tot de call gemaakt is alvorens je iets update. Zet gewoon één van de veranderende variabelen in een “computed property” en je problemen zijn opgelost.

Performance:

Met de one-page app die ik heb gemaakt, zijn er geen opmerkelijke verschillen in performance tussen de 2 frameworks, omdat het nu eenmaal een kleinschalige applicatie is.

Na wat opzoekingswerk , wordt het echter wel duidelijk dat Angular bij grotere applicaties traag kan worden. Als je wat online opzoekt, zie je toch een boel klachten van ontwikkelaars die vragen hebben bij de performance.

Velen zeggen dat het door de “dirty checking” ,“ HTML -parsing” en de” two-way data binding “komt en dat dit alles een groot nadeel is bij Angular.

Je vindt veel minder klachten over de performance bij Vue. Hier blijkt de performance niet te bezwijken bij het groter worden van de app. Hoewel Vue veel minder gebruikers heeft en er dus ook minder klachten over zullen verschijnen, is dit toch veelbelovend. Vue gebruikt ook veel dezelfde eigenschappen als Angular en toch blijkt de performance reuze mee te vallen.

Conclusie:

Beide frameworks hebben hun eigen voor- en nadelen en zijn beiden zeer nuttig voor het maken van bepaalde applicaties. Het structurele en overzichtelijke aspect van Angular zorgt ervoor dat het goed bruikbaar is voor grotere applicaties met verschillende directives.

Het “easy to learn” en compacte aspect van Vue doet vermoeden dat het zeer nuttig is voor kleinere projecten die geen enorm “design pattern” nodig hebben. Ik vermoed wel dat er nog een heleboel nieuwe tools en aanpassingen op stapel staan bij Vue en dat er dus in de toekomst nog meer mogelijkheden zullen bestaan.

Ik ben in elk geval een fan van beide frameworks: Ze maken het leven gemakkelijker en ik ben er zeker van dat ik beide frameworks nog veel zal gebruiken in de toekomst.