**Introductie React**

In dit artikel ga ik het framework React bespreken. React is een van de bekendste frameworks van dit moment., en ziet er naar uit dat als je als developer geen React beheerst, je eigenlijk niet meespeelt. Bij veel van de weekly mingles kwam React ter sprake, bijna elk bedrijf maakte gebruik van deze programmeertaal, dit is waarom ik als beginnend programmeur meer wil weten over React en daarom dit artikel ook schrijf.

**Herkomst van React**

Eerder noemde ik React een framework, dit terwijl dat eigenlijk helemaal niet zo is. React is namelijk een library die van origine door Facebook in 2013 is ontwikkelt. Dit is ontwikkelt omdat React het mogelijk maakt om een pagina te refreshen zonder dat er op refresh hoeft worden geklikt, perfect dus voor social media platformen om snel en gemakkelijk nieuwe content te laden.

**Voordelen van React**

React maakt het mogelijk om snel en gemakkelijk dynamische websites en apps te creëren. Wat React zo populair maakt is dat je snel en gemakkelijk zowel client side als server side code kan schrijven. Door het maken van componentes kan je in React heel gemakkelijk communiceren tussen de client en server side. Een ander voordeel is dat React gemakkelijk te lezen is en het dus ook makkelijker maakt om code te onderhouden. Door deze duidelijke structuur en goede feedback is het in React redelijk gemakkelijk om te debuggen en is het perfect als je in teamverband aan een project werkt.

**React voor apps**

React biedt ook nog eens verschillende extensies, een daarvan is React Native. Dit wordt gebruikt voor het ontwikkelen van mobiele apps wat dus de programmeer taal op elk vlak bruikbaar maakt. Sterker nog, React native wordt ondersteund door Apple, wat het dus mogelijk maakt om apps voor de Appstore te ontwikkelen zonder daarbij Swift te hoeven te leren.

**Componenten**

In React wordt er veel gebruik gemaakt van componentes, die dus gemakkelijk zowel client als server side kunnen worden opgehaald. Een component in React is éen van de delen van de user interface, deze components zijn gemakkelijk her te gebruiken wat zorgt voor snelle en duidelijke ontwikkeling. Een is gestructureerd als een object. Componenten kunnen in React verschillende states hebben, als een state verandert wordt dit component automatisch in de user interface verandert naar de juiste state.

**Conclusie**

Wat ik uit deze korte introductie kan concluderen is dat React een snel, innovatief en populair framework is die bij de gehele developer community bekent is. Naar mijn mening zit er veel toekomst in React en raadt ik elke developer aan om zich hierin te verdiepen.

Dit was een korte introductie over React. React is na het volledig onder de knie krijgen van Javascript de eerste library (nee geen framework) die ik wil gaan leren. Al met al lijkt React mij een super interessant onderwerp voor in de toekomst, omdat er ook heel veel potentie voor later in zit.

**Bron:**

* Introduction React:

<https://towardsdev.com/introduction-to-react-js-df06464ebc1b>

# The most important lessons I’ve learned after a year of working with React:

https://www.freecodecamp.org/news/mindset-lessons-from-a-year-with-react-1de862421981

**Mongoose**

In dit artikel ga ik het hebben over Mongoose. Ik schrijf dit artikel omdat ik zelf voor het eerst met een database heb gewerkt en ik zeer te spreken was over Mongoose. Om dit nog beter te begrijpen schrijf ik dit artikel als introductie voor Mongoose en Mongodb.

**Wat is Mongoose?**

Mongoose is een Javascript framework die samen werkt met Mongodb en Node.js. Mongodb is een noSQL database. Met mongoose kan je snel en gemakkelijk communiceren met database. Volgens Mongodb zelf is het een elegantere manier van communiceren.

Mongoose is een object data modeling library. Met deze library kan je gemakkelijk data schema’s en modellen aanmaken in een Javascript bestand en deze makkelijk manipuleren.

Hiervoor is Mongoose perfect.

**Schema**

Om de data juist gestructureerd te krijgen maakt Mongoose gebruik van Schema’s. Deze schema’s maak je in een normaal Javascript bestand aan en deze worden uiteindelijk in een model geplaats, meer hier later in het artikel over. Een schema is zeer gemakkelijk op te stellen en maakt het ook mogelijk om data te nesten. Om een schema aan te maken moet je eerst mongoose ophalen in het Javascript bestand en daarna een constant variabele aanmaken voor het Schema, het is gebruikelijk om met een hoofdletter te beginnen. Nadat je dit hebt gedaan kun je daadwerkelijk de verschillende keys die je nodig hebt erin zetten. Elke key moet je een type meegeven, dit kan een array, boolean of bijvoorbeeld een number zijn. Hieronder een lijst met alle type’s:

* String
* Number
* Date
* Buffer
* Mixed
* ObjectId
* Array
* Map
* String
* Number
* Date

Naast het meegeven van een type, kan je ook nog aangeven of een key verplicht is om data in te zetten. Dit doe je door middel van required: boolean. Als dit gedaan is dan is je schema klaar en kan deze in een model worden gezet.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

**Model**

Een model zorgt ervoor dat het schema wordt toegepast bij alle documenten in de collectie. In je app.js is het belangrijk dat je een constante variabele maakt van dit model. De naam van het model moet altijd de enkelvoudige vorm van de naam van de collection op Mongodb. Om het variabelen aan te maken gebruik je deze code:

Afbeelding met tekst

Automatisch gegenereerde beschrijving

“Hierbij is Blog de enkelvoud van blogs, de titel van de collectie in de Mongodb database.”

**Met Mongodb verbinden**

Als de gehele structuur af is, dan is het belangrijk dat er een connectie wordt gemaakt met de Mongodb database. Hier is eigenlijk altijd een standaard code voor. Je moet alleen nog je gebruikersnaam en wachtwoord invullen. Zorg altijd dat deze prive zijn!

const dbURI = 'mongodb+srv://username:password.oyxfpmr.mongodb.net/VacaturebankData?retryWrites=true&w=majority';

mongoose.connect(dbURI, { useNewUrlParser: true, useUnifiedTopology: true })

.then((result) => app.listen(2000), console.log('Mongodb connected'))

.catch((error) => console.log(error + 'has occured'))

**Data toevoegen**

Het is zeer gemakkelijk om in Mongoose data toe te voegen. Eerst maak je een constante variabele aan, vaak met een naam die gerelateerd is aan de data die je gaat opslaan. Je creeert dan een nieuw model en schrijft alle waardes erin. Bij een formulier in Node.js wordt dit bijvoorbeeld opgehaald als volgt:

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Door dit kleine stukje code wordt er data opgeslagen op Mongodb, hier zie je ook hoe gemakkelijk het is om Mongoose te gebruiken.

**Data updaten**

Mongoose maakt het heel gemakkelijk om dynamisch data te veranderen. Dit doe je simpelweg door de variabele opnieuw te schrijven en alleen de waarde te veranderen.

Afbeelding met tekst

Automatisch gegenereerde beschrijving

**Het zoeken van data**

Het zoeken van data is in Mongoose relatief gemakkelijk. De meest bekende is findById, wat het mogelijk maakt om via het ObjectId specifieke data te vinden. Afbeelding met tekst

Automatisch gegenereerde beschrijving

**Conclusie Mongoose**

Dit is hoe ver ik ben in het leren van Mongoose, er is nog een stuk meer te leren over het manipuleren van data. Hier ben ik nu druk mee bezig bij het ontwikkelen van mijn eindproject. Al met al vind ik Mongoose super fijn werken, het zorgt voor een snelle en dynamische interactie tussen de applicatie en de database. Een min puntje waar ik al meerdere dagen mee vast zit, is dat het lijkt alsof je geen extra key kan toevoegen in een bestaand schema, hierdoor kan ik geen favorieten aan de gebruiker koppelen. Ik hoop dit de komende dagen te leren en onder de knie te krijgen. Wel raad ik iedereen Mongoose aan, alleen zorg er wel voor dat je het schema vooraf helemaal uitwerkt en structureert zodat je niet tegen hetzelfde probleem als ik aanloop.

**Bronnen:**

* Getting started with mongoose:

<https://www.mongodb.com/developer/languages/javascript/getting-started-with-mongodb-and-mongoose/>

* Eigen ervaring

**Introductie Typescript**

Bij veel van de weekly mingles kregen wij vaak te horen dat ze Typescript gebruikten in plaats van Javascript. Daarom wil ik in dit artikel kort bespreken wat typescript is en wat zijn voordelen zijn tegenover het welbekende Javascript.

**Wat is typescript?**

Typescript is eigenlijk een extra geavanceerde laag op Javascript. Javascript is een zeer dynamische taal die eigenlijk weinig restricties heeft. Zo kan je gemakkelijk de waarde van een variabele van een string naar een nummer veranderen. Typescript zorgt dat je wel gemakkelijk restricties kan aanmaken, het zal dan bijvoorbeeld niet mogelijk zijn om het type waarde van een variabele zomaar te veranderen. Dit zal een foutmelding opleveren. Tevens is Typescript een stuk specifieker in het geven van error meldingen, wat debuggen voor developers een stuk gemakkelijker kan maken.

**Voordelen van Typescript**

Het grootste voordeel van Typescript is dat je gemakkelijk restricties kan toevoegen wat programmeren iets overzichtelijker zou kunnen maken. Wat vooral interessant is, is dat met Typescript beginnen zeer makkelijk is. Dit komt omdat elke Typescipt code valide code in normaal Javascript is en andersom. Dit maakt het voor developers een stuk makkelijker om te experimenteren met Typescript in bestaande Javascript projecten. Wat fijn is aan Typescript is dat je variabelen een type kan meegeven zoals een boolean, string, array etc.

Door hier goed gebruik van te maken kan je ervoor zorgen dat je code sterk in elkaar zit en niet gevoelig is voor bugs. Typescript maakt het ook mogelijk om makkelijk toegang te krijgen tot opgeslagen componenten om deze snel en efficiënt te kunnen gebruiken.

**Conclusie**

Mijn conclusie is dat Typescript eigenlijk alleen maar voordelen met zich meeneemt. Typescript lijkt op de meest efficiente versie van Javascript en is naar mijn mening echt een opkomende programmeertaal. Ik kan concluderen dat het zeker de moeite waard is om jezelf in Typescript te verdiepen, dit omdat je dit later kan helpen om sneller, efficiënter en beter gestrucutreerde code kan schrijven.