Inhoud

1 Hoofdstuk

Kennismaken met Node.js

JavaScript is overal. Al jarenlang wordt gebruik gemaakt van JavaScript-bibliotheken en -frameworks in de browser. Erg bekend zijn onder meer jQuery, AngularJS en React. Een relatief nieuwe ontwikkeling is het gebruik van JavaScript in serversided projecten. Hiervoor wordt Node.js ingezet. Sinds het debuut van Node in 2009 is het een van de populairste projecten op Github, zijn er duizenden modules gepubliceerd voor de Node Package Manager (NPM) en is Node.js verplichte kost als u gebruik wilt maken van tools als PhoneGap, Grunt, Gulp, Bower en talloze andere. In dit hoofdstuk maakt u kennis met enkele achtergronden van Node.js en kijken we naar het inrichten van een Node.js-werkomgeving.

In dit hoofdstuk:

Wat is Node.js?

Waarom Node.js gebruiken?

Kenmerken van Node.js-applicaties.

De werkomgeving instellen.

Plaats van dit boek in de Web Development Library.

Wat is Node.js?

Als u het vakgebied van de web developer een beetje gevolgd hebt de afgelopen jaren, zal het u niet ontgaan zijn: JavaScript is overal. Hoewel JavaScript dit jaar zijn twintigste verjaardag viert (de taal is in 1995gemaakt door Brendan Eich), is JavaScript pas echt populair geworden door de introductie van jQuery in 2006. Daarna is het hard gegaan. Er zijn ondertussen talloze jQuery-plug-ins beschikbaar. Ook op het gebied van complete JavaScript-frameworks zijn grote stappen gezet door onder meer Durandal, Ember, AngularJS en meer recentelijk Angular2, Aurelia en React. Gemeenschappelijk kenmerk van al deze tools is dat ze in de browser draaien.

De kwaliteit van JavaScript

We doen in dit boek geen uitspraken over de kwaliteit van JavaScript als programmeertaal. Of u nu van JavaScript houdt of niet, het is nu eenmaal de populairste programmeertaal op internet en de enige taal die draait in de browser. En met de introductie van Node.js dus ook op de server. De volgende versie van JavaScript (EcmaScript 2015, voorheen EcmaScript6, gestandaardiseerd in juni 2015) zal veel tekortkomingen van het huidige JavaScript aanpakken. Maar op dit moment is EcmaScript 2015 nog niet klaar voor productie. We gebruiken in dit boek dan ook traditioneel JavaScript (ES5), zoals het op dit moment door de Node V8-engine wordt ondersteund.

In dit hoofdstuk leest u meer over de belangrijkste achtergronden en kenmerken van Node.js. Met Node.js kunt u ook JavaScript schrijven in server-sided projecten. Wilt u direct in het diepe springen, ga dan aan de slag met de praktijk van hoofdstuk 2!

Server-sided JavaScript

Een gezamenlijk kenmerk van alle libraries die hiervoor werden genoemd is dus dat ze draaien in de browser. Het zijn client-sided frameworks. Node.js is dat niet. Node maakt JavaScript-applicaties beschikbaar buiten de browser. Op de website (**nodejs.org**) wordt Node.js omschreven als:

"Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices."

Praktijkgericht

Zoals in alle titels in de Web Development Library gaan we zo snel mogelijk in op de praktijk, zonder na te streven 100% volledig te zijn. Hiervoor verwijzen we graag naar de officiële documentatie en naar de talloze URL's in dit boek als achtergrondinformatie.

Ryan Dahl

Ryan Dahl van het bedrijf Joyent (**www.joyent.com**) is de oorspronkelijke maker van Node.js. Hij heeft van 2009 tot 2012 leiding gegeven aan het ontwikkelteam. Op dit moment is Timothy J. Fontaine de *project lead* van Node.js en is de verdere ontwikkeling in handen van de onafhankelijks stichting Node.js Foundation. Het bedrijf Joyent is nog steeds een van de sponsors van de Node.js Foundation, maar inmiddels wordt Node ook ondersteund door andere grote partijen in de markt (IBM, Microsoft, PayPal en andere).

Afbeelding 1.1 *Ryan Dahl stond aan de wieg van Node.js. Inmiddels is de ontwikkeling ervan in handen van de Node.js Foundation.*

De rol van io.js

Medio 2014 was er een scheuring in de Node.js-wereld. Een aantal personen vond dat de ontwikkeling van Node.js de verkeerde kant op ging en ze richten daarom io.js op als alternatief. Sinds mei 2015 zijn de gelederen weer gesloten en werken beide partijen samen in de Node.js Foundation. Er is nu gelukkig dus weer één versie van Node.js.

Kenmerken van Node.js

We zijn zo vrij de definitie van het Node.js-team beknopt uit te leggen en als volgt te vertalen:

- Node.js is een standalone versie van de JavaScript-runtime die in Google Chrome aanwezig is. Of, zo u wilt: de JavaScript-engine is losgetrokken uit Chrome en als standalone product

beschikbaar gemaakt. Zo kunnen JavaScript-apps buiten de browser draaien.

- Node maakt hiervoor gebruik van een event-driven architectuur. Dit betekent dat u veel code zult schrijven die reageert op events. Callbackfuncties liggen ten grondslag aan de werking van vrijwel elke Node-applicatie.
- Node is gemaakt voor het bouwen van snelle, schaalbare netwerkapplicaties. Dat betekent dat de module `http` is ingebouwd. Er is dus standaard een netwerkprotocol aanwezig. De module `http` is een zogenoemde *first class citizen* in Node.js-applicaties. Er is geen andere webserver nodig zoals Apache, nginx of Internet Information Server.
- Omdat een programma alleen maar hoeft te reageren op events, is de performance onwaarschijnlijk goed. Nooit wordt de uitvoering geblokkeerd omdat het programma op het resultaat van een berekening staat te wachten. In plaats daarvan wordt de berekening uitbesteed aan een functie of andere externe toepassing (bijvoorbeeld het ophalen van data uit een database). Er wordt een callbackfunctie uitgevoerd als de berekening gereed is, of de data is geladen. Dit wordt bedoeld met het *non-blocking I/O model* in de beschrijving. In pseudocode zou dat er als volgt uitzien:

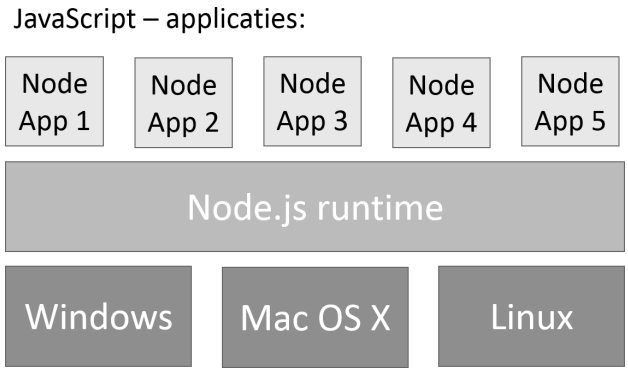
```
// ***** FOUT: synchrone code
var user = User.findById(100);
// ...wacht tot operatie gereed is, doe daarna iets met user.
```

```
// ***** GOED: asynchrone code
User.findById(100, function(user){
    // callback: doe iets met user zodra de operatie compleet is.
});
```

- Wat blijkt nu? Op het web vindt enorm veel I/O-verkeer plaats. Ga maar na: elke bestandshandeling (het serveren van HTML-, CSS- en JavaScript-bestanden) is een I/O handeling. Het ophalen of wegschrijven van gegevens zijn I/O-operaties

en ga zo maar door. Daarom is Node.js enorm geschikt voor webservers of realtime, data-intensieve handelingen.

- Node.js is geschreven *voor* JavaScript, het is niet geschreven *in* JavaScript. Node.js zelf is gewoon een uitvoerbaar programma dat is geschreven in C++ en geschikt is gemaakt voor Windows, Mac en Linux. Op die systemen moet het apart worden geïnstalleerd. Maar omdat alle Node-toepassingen zelf zijn geschreven in JavaScript, hoeft u het maar één keer te schrijven en draait de applicatie vervolgens (via Node) op alle besturingssystemen. Deze architectuur is te zien in afbeelding 1.2.



Afbeelding 1.2 *Node.js is als runtime beschikbaar voor Windows, Mac OS X en Linux. De toepassingen schrijft u in JavaScript.*

- Node.js maakt gebruik van één JavaScript-engine, de Chrome V8-engine. Dit betekent (hoera!) dat u in applicaties geen rekening meer hoeft te houden met verschillende JavaScript-dialecten of afwijkende implementaties in verschillende browsers. Zowel op Windows, Mac als Linux programmeert u voor dezelfde engine. Node-JavaScript is altijd en overal hetzelfde. Een abstractie-library als jQuery zult u voor Node.js dan ook nooit nodig hebben.

De JavaScript event loop

Als u nog niet zo bekend bent met de JavaScript event loop of de manier waarop JavaScript intern omgaat met asynchrone code, bekijk dan de presentatie van Philip Roberts op YouTube: *what the heck is the event loop anyway?*, op **www.youtube.com/watch?v=8aGhZQkoFbQ**.

Afbeelding 1.3 *JavaScript is een event driven taal. Bekijk deze presentatie om te zien wat dit exact betekent.*

Geen DOM beschikbaar

Een ander kenmerk van Node-toepassingen is dat ze geen traditioneel DOM kennen zoals u gebruikt in jQuery, AngularJS en alle andere JavaScript-tools waarmee u bekend bent. Immers, de applicatie draait buiten de browser. Dit betekent ook dat u geen opdrachten kunt gebruiken als `document.getElementById()` of `alert()`. Dit zijn zoals u weet allemaal DOM-handelingen.

In plaats daarvan zult u vaak `console.log()`-meldingen schrijven of in een debugger gebruik maken van watches of breekpunten om code te verbeteren. Hier gaan we later in het boek nog uitgebreid op in.

JSON

Als u een applicatie data wilt laten retourneren (u schrijft bijvoorbeeld een Node.js API), dan wordt bijna altijd het JSON-formaat gebruikt. Het volgende statement is bijvoorbeeld een geldige Node.js-opdracht om data terug te geven vanuit een functie:

```
return ({"naam" : "Peter Kassenaar"});
```

Ook databases die vaak worden gebruikt in combinatie met Node.js zoals MongoDB of CouchDB retourneren hun informatie meestal in het JSON-formaat. Kortom: zorg er voor dat u JSON kent en begrijpt als u met Node.js aan de slag gaat. Meer informatie is desgewenst beschikbaar op **http://json.org**.

Ook andere databases

Vaak worden zogenoemde NoSQL-databases als CouchDB of MongoDB ingezet in Node-toepassingen. Maar het is zeker mogelijk om Node-applicaties te laten praten met traditionele databases zoals MySQL of SQL Server. Daarvoor is dan wel altijd een extra driver nodig. In hoofdstuk 7 gaan we in op het werken met MongoDB.

De MEAN-stack

Als u bezig gaat met Node.js-toepassingen, zult u ongetwijfeld het begrip *MEAN-stack* tegenkomen. Dit is een bundeling van vier verschillende technieken en modules die gezamenlijk worden ingezet om (web-)applicaties te schrijven. De vier letters staan voor de tools die worden gebruikt:

- **M voor MongoDB** De database voor een applicatie.

- **E voor Express** De module die wordt ingezet om de webserver, routing en API te maken.
- **A voor AngularJS** Het framework waarmee de website wordt gemaakt.
- **N voor Node.js** De motor voor zowel MongoDB als Express.

Hierbij is alleen AngularJS een client-sided component. De rest draait allemaal aan de serverkant. Node.js en Express *zijn* de server, MongoDB is de database die wordt aangesproken vanuit Node.js en Express.

In dit boek maken we gebruik van alle componenten van de MEAN-stack. Na afloop kunt u zich daarom zeker een beeld vormen van het type applicaties dat hiermee wordt gemaakt. U kunt zelf MEAN-applicaties schrijven. De ruimte schiet te kort om uitgebreid in te gaan op AngularJS (zie ook verderop bij Tips voor meer lezen), maar we gebruiken het wel om in de flow van de totale stack te blijven.

Waarom Node.js gebruiken?

Het zou verkeerd zijn om Node.js te gebruiken “omdat iedereen het doet” of “omdat het een coole, nieuwe techniek is” (terzijde: de eerste versie van Node.js stamt alweer uit 2009. Zo vreselijk nieuw is het dus ook weer niet…). U moet zich goed realiseren wat het gebruik van Node.js betekent. De belangrijkste overwegingen zijn:

- **Eén programmeertaal: JavaScript in de client en op de server** U hoeft niet meer te investeren in kennis van zowel een front-end-techniek (JavaScript) als een backend-techniek (Java, C#, PHP enzovoort). Alles is JavaScript. U hoeft nog maar één programmeertaal te leren en de kennis hieromtrent te onderhouden.

- **Code opnieuw gebruiken** Omdat zowel op de server als op de client dezelfde taal wordt gesproken, kan code worden hergebruikt. Het belang hiervan moet overigens niet overdreven worden. Sommige logica zult u misschien opnieuw kunnen gebruiken, maar alle code die met de webpagina (het DOM) te maken heeft, zult u apart moeten schrijven.
- **Snellere ontwikkeltijd** Omdat niet meer geschakeld hoeft te worden tussen verschillende programmeertalen, kan sneller worden ontwikkeld. Node.js-applicaties hoeven ook niet gecompileerd en/of uitgerold (*deployed*) te worden. Net zoals in een website met HTML, CSS en JavaScript kunt u gewijzigde code meestal direct opnieuw testen. Er zijn verschillende live reload- of monitoring-tools aanwezig die een Node-server opnieuw starten zodra ze zien dat een bestand is gewijzigd.
- Veel JavaScript-kennis aanwezig Er zijn al duizenden programmeurs opgeleid die erg vaardig zijn met client-sided JavaScript. Deze kennis kunnen ze nu inzetten in een server-sided omgeving. Uiteraard zijn er nieuwe concepten die moeten worden geleerd. In Node.js kunt u bijvoorbeeld werken met het bestandssysteem via de module `fs`, in de browser kan dat niet. En zo zijn er nog een aantal verschillen.
- **Ondersteuning van uit community** Er zijn talloze JavaScript-blogs. Duizenden vragen op Stackoverflow.com en Quora behandelen JavaScript-aspecten. Node.js is een van de populairste repositories op Github. Er zijn meer dan 100.000 open source packages beschikbaar voor NPM. Met andere woorden: het moet wel erg raar lopen als u tegen een probleem oploopt waarvoor vanuit de community nog geen antwoord beschikbaar is.

Wanneer Node.js niet gebruiken?

Node.js is natuurlijk niet geschikt voor alle doeleinden. Als u een van de volgende doelstellingen hebt kunt u voorlopig misschien beter naar een andere tool omkijken.

- **Snelle website-ontwikkeling** Node.js is geen WordPress, Drupal of Joomla. Hoewel er zeker CMS'en en frameworks beschikbaar zijn voor kant-en-klare websites (we noemen bijvoorbeeld **keystonejs.com** of **pencilblue.org**), is de ontwikkeling hiervan nog niet op hetzelfde niveau zoals u kent van de bekende spelers in deze markt.
- **CPU-intensieve applicaties** De JavaScript V8-engine waar Node.js op is gebouwd is supersnel, maar als u zeer rekenintensieve taken wilt uitvoeren (beeldbewerking, audio- en videotooling), kijk dan liever nog even verder naar native tools hiervoor. Node.js is vooral geschikt voor realtime communicatie, intensieve bestands-I/O en event driven toepassingen.

Verschillende manieren om Node.js te gebruiken

Veel gebruikers hebben (vaak zonder dat ze zich hier van bewust zijn) Node.js ooit al eens geïnstalleerd op hun computer. Dit omdat ze werken met PhoneGap, de JavaScript task runner Grunt, het testframework Karma of andere producten. Dit zijn allemaal Node modules. Om deze te kunnen installeren en gebruiken, moeten Node.js en de bijbehorende package manager NPM dus geïnstalleerd zijn. Misschien geldt dit ook voor u.

In dat geval gebruikt u Node alleen maar als platform om andere tools te kunnen gebruiken. U doet eigenlijk niks met Node zelf. Maar daar gaan we in dit boek verandering in aanbrengen!

Globaal gesproken zijn er dus twee manieren om Node.js te gebruiken:

1. Gebruik Node als platform om bestaande modules (PhoneGap, Bower, Grunt enzovoort) te kunnen installeren en gebruiken. Node.js is in dat geval een vereiste om de modules te kunnen gebruiken, maar er wordt verder niet rechtstreeks iets mee gedaan.

2 Gebruik Node als platform om zelf applicaties voor te ontwikkelen. Dit is de weg die we in dit boek bewandelen. U maakt in dit geval echt gebruik van Node zelf en schrijft met JavaScript toepassingen om eigen modules te ontwikkelen zoals webservers, communicatie met het bestandssysteem en databases en meer.

Benodigde voorkennis

Dit boek maakt deel uit van de *Web Development Library* (**www.webdevelopmentlibrary.nl**). In elk deel wordt een op zichzelf staande techniek besproken die te maken heeft met webdevelopment. Andere, gerelateerde technieken worden bekend verondersteld. Zo betaalt u alleen voor datgene wat u echt nodig heeft.

- In dit boek gaan we in op Node.js 0.12.4. Het is een server-sided JavaScript-runtime. We gaan er dan ook van uit dat u voldoende ervaring hebt met JavaScript. Variabelen, statements, functies en objecten worden niet apart besproken. Er wordt redelijk diepgaande kennis bekend verondersteld van JavaScript-concepten.
- U moet kunnen werken in een opdrachtregelomgeving of Command Line Interface (*CLI*). Dit is cmd.exe in Windows en Terminal in Mac en Linux. Hierin moet u de belangrijkste opdrachten kennen zoals `dir`/`ls`, `md`/`mkdir`, `cd` enzovoort.
- Kennis van de Engelse taal is erg handig. Veel video's, informatie en blogposts zijn alleen in het Engels beschikbaar. Hier wordt in het boek regelmatig naar verwezen.