Inhoud

1 Hoofdstuk

Kennismaken met Vue.js

Van enkele eenvoudige HTML-pagina's in de jaren 1990 is het web uitgegroeid tot een van de meest complexe systemen die we kennen. Internet wordt gebruikt voor eenvoudige hobbysites, maar ook voor onlinebetaalsystemen, crm- en klantbeheersystemen, verzekerings- en schademodellen, sociale media en ontelbare andere zaken. Vue.js is een framework voor het programmeren van dergelijke ingewikkelde webapps. In Vue.js werkt u niet meer met losse HTML-pagina's, maar met webcomponenten. De componenten werken met elkaar samen en vormen zo een complete webapplicatie. Vue.js maakt het mogelijk in hoge mate modulair te programmeren. JavaScript is de programmeertaal die hiervoor wordt ingezet. Dit boek geeft een inleiding op al deze zaken. Na afloop kunt u vol vertrouwen aan de slag met uw eigen Vue-applicaties.

In dit hoofdstuk:

Wat Vue.js is, en wat het niet is.

Concepten en kenmerken van Vue.

Benodigde voorkennis en software.

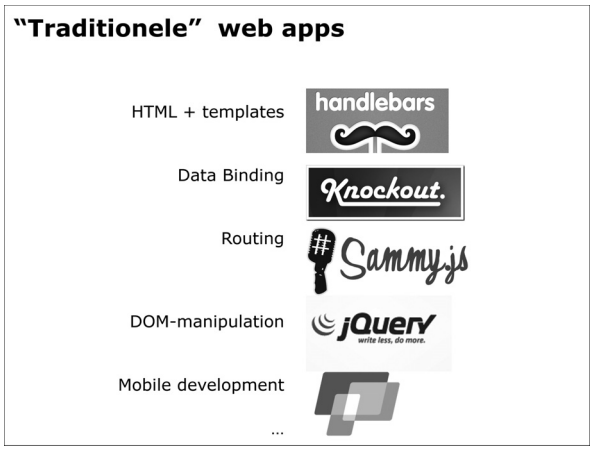
Wat hebt u nodig? De werkomgeving inrichten.

Wat is Vue.js?

Het aloude HTML is prima om tekst en afbeeldingen te tonen in de browser, maar is oorspronkelijk nooit ontwikkeld voor het maken van dynamische webapplicaties. Voor dat doel is JavaScript rond 1995 ontworpen. Samen met CSS (dat rond dezelfde tijd opkwam) behoort JavaScript op dit moment tot de basisvaardigheden van elke webdeveloper. JavaScript was in het begin lastig te leren en verschillende browsers hadden hun eigen ideeën over de implementatie van JavaScript.

Libraries en frameworks

Pas sinds de opkomst van aanvullende bibliotheken zoals jQuery in 2006 heeft JavaScript een enorme vlucht genomen. Behalve jQuery zijn tal van andere bibliotheken (*library's*) ontwikkeld, elk met hun eigen doel. Er zijn bibliotheken voor DOM-manipulatie (zoals jQuery), routing (sammy.js), databinding (backbone, knockout.js), werken met datums en tijden (moment.js) en nog veel meer.



Afbeelding 1.1 *In traditionele webapplicaties neemt elke bibliotheek een van de eisen die aan de applicatie wordt gesteld voor zijn rekening. Er is kans op onderlinge incompatibiliteit.*

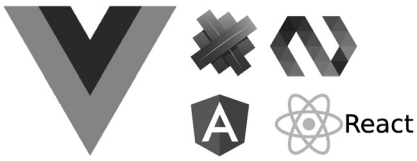
Maar Vue is geen bibliotheek zoals de hiervoor genoemde. Vue is een compleet *framework* voor het realiseren van clientside webapplicaties. Een framework vervult *alle* taken die aan moderne webapplicaties worden gesteld.

- Als we de zaken erg vereenvoudigd voorstellen, kun je zeggen dat bibliotheken in het algemeen één ding heel goed doen.
- Een framework zoals Vue.js, biedt oplossingen voor alle niveaus van applicatieontwikkeling. Van het structureren en binden van data tot Ajax-communicatie met webservers, het verwerken van geretourneerde gegevens in de HTML-template en het maken van herbruikbare componenten.

Bibliotheken kunnen in het algemeen gecombineerd worden in een project om gezamenlijk het beste resultaat te bereiken. Bij frameworks maak je daarentegen één keuze en bouw je de app volgens de richtlijnen en kenmerken van het gekozen framework.

Geen combinaties

We zullen in de praktijk bijvoorbeeld nooit zien dat een app *zowel* Vue.js als React (een alternatief framework) gebruikt. Ook combinaties van Vue met Angular of Polymer (andere alternatieven) komen niet voor. U bouwt de site ofwel in Vue, ofwel in Angular. Niet in beide.



Afbeelding 1.2 *In een framework als Vue.js, maar ook in alternatieven zoals Angular, Polymer of Aurelia, worden alle taken van een applicatie gebundeld en geïntegreerd aangeboden. De leercurve is steiler, maar het resultaat is een consistentere en eenvoudigere (en dus goedkoper te onderhouden) applicatie.*

Omschrijving van Vue.js

Vue.js wordt op de officiële site (**www.vuejs.org**) omschreven als:

The progressive JavaScript framework

Hiermee wordt bedoeld dat, hoewel Vue.js als één framework wordt aangeboden, het in beginsel een kleine kern heeft. Deze kern kan naar behoeven worden uitgebreid ('progressief'). Hiervoor gebruikt Vue aanvullende bibliotheken. U mág ze inzetten, maar het hoeft niet. De kernbibliotheek is voornamelijk gericht op de weergavelaag (*view layer*) van applicaties en richt zich met name op HTML-templates en data die daarin wordt weergegeven.

Progressieve ontwikkeling

Als de applicatie meer nodig heeft, zoals routeren naar verschillende componenten of het communiceren met een backend, kunt u daarvoor aanvullende bibliotheken inzetten. Deze zijn vaak speciaal voor Vue.js ontworpen, maar ze zijn niet verplicht om een werkende Vue-applicatie te maken.

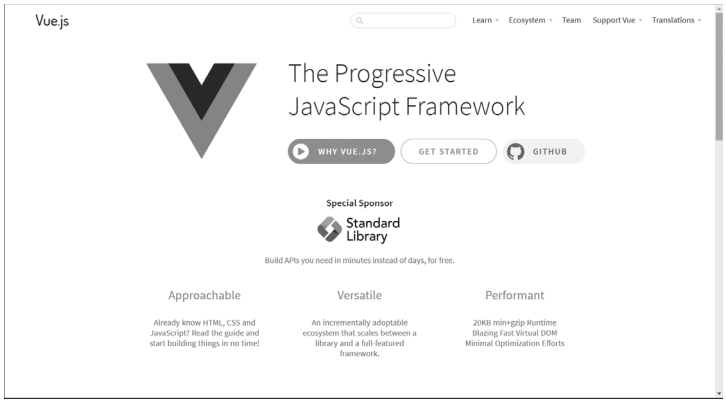
Soms zijn de aanvullingen ook gemaakt door andere ontwikkelaars en geadopteerd in het Vue.js-framework. Bekende aanvullende bibliotheken zijn bijvoorbeeld:

- **Vue Router** voor het routeren in applicaties van de ene component naar de andere component, waarbij de URL in de adresbalk van de browser wordt aangepast;
- **Axios** voor het communiceren met API's om gegevens/data op te halen en vervolgens te tonen in de applicatie;
- **Vuex** een oplossing voor state management, om alle data in de applicatie op een centrale plek te beheren;

- **Vue CLI** de opdrachtregelomgeving voor het instellen van projecten, het regelen van afhankelijkheden (*dependencies*), testen en meer;
- **Vue.js devtools** een uitbreiding voor Google Chrome en Firefox die het inspecteren en debuggen van Vue-applicaties vereenvoudigt.

Er zijn er uiteraard meer, maar dit zijn de bibliotheken waar u als Vue.js-ontwikkelaar ongetwijfeld mee te maken krijgt. Lees eventueel meer over de kenmerken van Vue.js op **vuejs.org/v2/guide**.

Vue.js is een compleet clientside framework. Het is in JavaScript geschreven en draait ook volledig in de browser. In de ideale situatie is de app compleet losgekoppeld van de server en database waar de gegevens vandaan komen. Alle communicatie vindt plaats via Ajax-aanroepen. Uiteraard gaan we hier later in dit boek nog dieper op in. Om Vue.js-applicaties te maken hebt u geen enkele kennis nodig van een backendtechnologie zoals PHP, Java of C#.



Afbeelding 1.3 *De homepage van Vue op vuejs.org. Start hier voor officiële downloads, documentatie en meer.*

Vue of Vue.js?

De begrippen Vue en Vue.js worden door elkaar gebruikt. De officiële en volledige naam is Vue.js, maar in het algemeen spraakgebruik wordt vaak kortweg gesproken van Vue. In dit boek bedoelen we er hetzelfde mee: uw applicatie die draait in de browser en is geschreven in het framework Vue.js.

Vue op internet

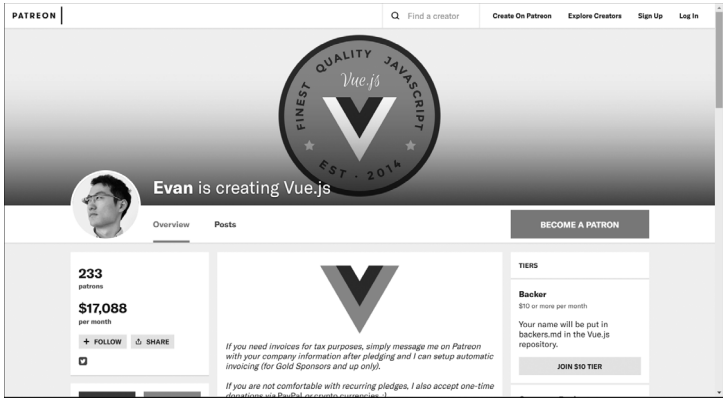
De homepage van Vue is te vinden op **vuejs.org**. Hier kunt u artikelen lezen, online lessen volgen, deelnemen aan discussies en zelfs Vue-T-shirts en andere merchandise aanschaffen. Ook is dit het startpunt voor de officiële documentatie. Kies hiervoor de optie **Learn**, **Guide** uit het hoofdmenu.

Geen groot bedrijf

In tegenstelling tot andere frameworks staat er achter Vue.js geen groot bedrijf. Angular is gemaakt (en wordt financieel onderhouden) door Google. React is afkomstig uit de stal van Facebook en TypeScript is oorspronkelijk gemaakt door Microsoft. Maar Vue.js is in beginsel een eenmansproject van Evan You (@youyuxi op Twitter), al werken er op dit moment veel meer mensen aan Vue.js. Dit is mogelijk gemaakt door middel van donaties en contributies uit de opensourcegemeenschap. Vue.js is gestart in 2014 en op het moment van schrijven dus ruim vijf jaar oud.

Maakt uw bedrijf of organisatie ook gebruik van Vue.js, overweeg dan een donatie aan het Vue-project via Patreon. De site is te vin-

den op **www.patreon.com/evanyou**. Zo ondersteunt u de makers en kunnen zij zich bezig houden met verbeteringen, het repareren van fouten en meer.



Afbeelding 1.4 *Vue.js wordt volledig gefinancierd door donaties uit de opensourcegemeenschap. Het maandbedrag wordt verdeeld onder alle ontwikkelaars. Overweeg een gift als u Vue.js in uw organisatie gebruikt.*

Dit boek ondersteunt Vue.js

Door dit boek te kopen, hebt u Vue.js indirect al ondersteund. Van de opbrengsten van dit boek maakt de auteur namelijk 15% over aan de Vue.js-organisatie. Ook een deel van de opbrengsten uit de Vue.js-trainingen die de auteur organiseert, worden gedoneerd om verdere ontwikkeling van Vue mogelijk te maken.

Populariteit van Vue.js

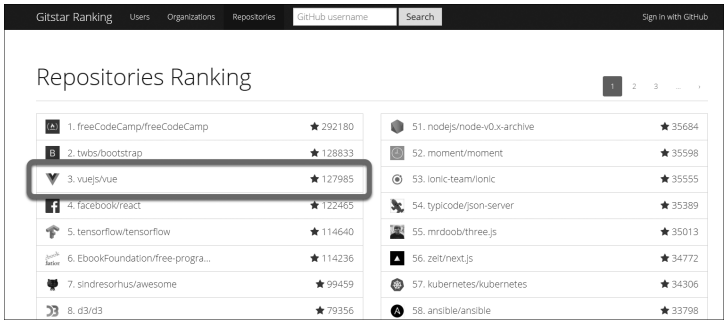
Er zijn verschillende manieren om de populariteit te meten, en geen van de manieren is de enige juiste wijze. Gecombineerd laten ze echter het volgende beeld zien. Op het moment van schrijven van dit boek (zomer 2019) is Vue.js in omvang en popu-

lariteit het derde framework. React is het grootst/populairst, gevolgd door Angular. De frameworks Polymer en Aurelia worden verreweg het minst gebruikt en zijn niet meegenomen in de afbeeldingen.

Github sterren

Op **github.com** (waar alle populaire opensourcerepositories zijn gehost) kunnen ontwikkelaars een ster geven aan een repository om zo een lijstje met favoriete repositories bij te houden. Hierbij zien we dat Vue.js het populairst is, met bijna 130.000 sterren:

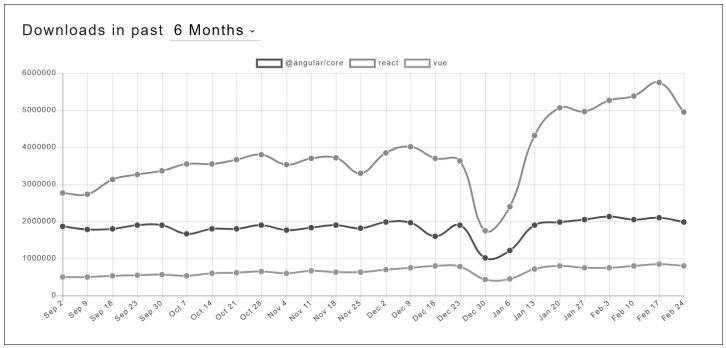
1. Vue.js 127.000 sterren
2. React 122.000 sterren
3. Angular 42.000 sterren



Afbeelding 1.5 *Vue.js staat wereldwijd op de derde plaats van populaire repositories en is het populairste frontend framework bij ontwikkelaars. Bekijk zelf de huidige positie op gitstar-ranking.com/repositories.*

Npm downloads

Wanneer de opdracht `npm install` voor een project wordt uitgevoerd, wordt de package opgehaald uit het npm-register. Dit zult u zelf nog talloze malen doen als u de oefeningen in dit boek volgt of de voorbeeldprojecten download en uitvoert. De website **npmtrends.com** houdt bij hoe vaak een bepaalde package wordt

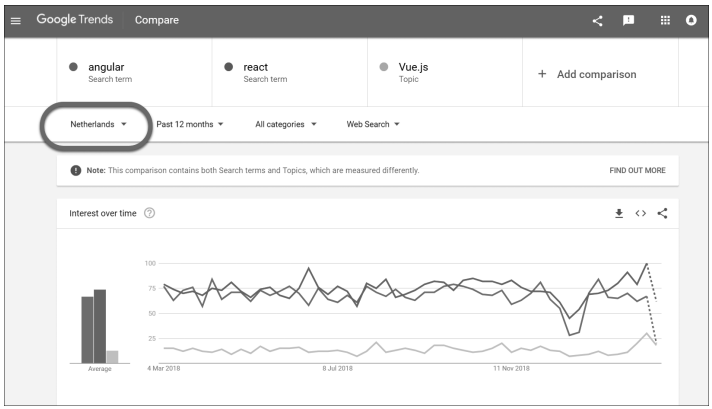


Afbeelding 1.6 *Een vergelijking in aantallen downloads voor React, Angular en Vue bij npmtrends.com.*

geïnstalleerd. Hier is duidelijk te zien dat React en Angular het meest worden geïnstalleerd en Vue op de derde plaats staat.

Google Trends

Tot slot Google Trends (**trends.google.com**). Hiermee is te onderzoeken hoe vaak een zoekvraag wordt gebruikt in relatie tot andere zoekvragen. Ook dan zien we min of meer hetzelfde beeld. React en Angular staan op de plekken één en twee, Vue is nummer drie. Boze tongen zullen overigens beweren dat React



Afbeelding 1.7 *De interesse bij Google voor de drie grote frameworks, op een rijtje gezet door Google Trends. De regio Netherlands is geselecteerd. Zelf kunt u ook allerlei andere filters toepassen.*

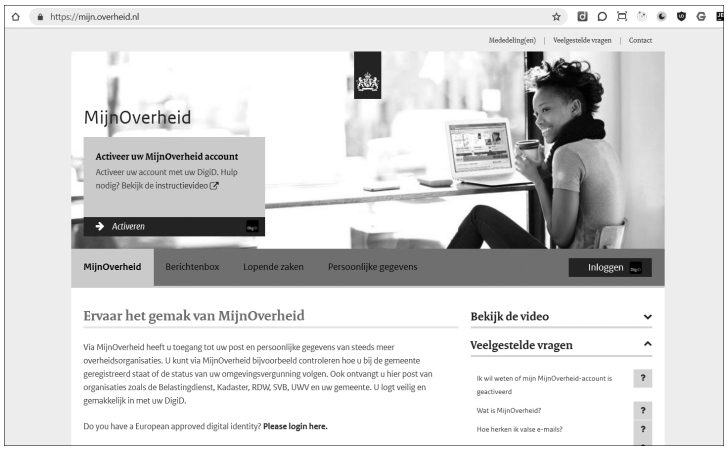
bovenaan staat omdat het zo verduiveld moeilijk is om te leren en er dus veel zoekvragen voor nodig zijn. Vue.js daarentegen is een stuk eenvoudiger en staat dus logischerwijs op plek drie…

Het algemene beeld van al deze vergelijkingen is echter duidelijk. React is het grootste, gevolgd door Angular. Vue.js wint echter aanzienlijk aan populariteit en kent een stijgende lijn. Het is niet ondenkbaar dat in de toekomst React of Angular van de troon wordt gestoten door Vue.

Wie gebruiken Vue.js?

Inmiddels wordt Vue.js door allerlei bedrijven, van klein tot (zeer) groot ingezet. Enkele voorbeelden zijn:

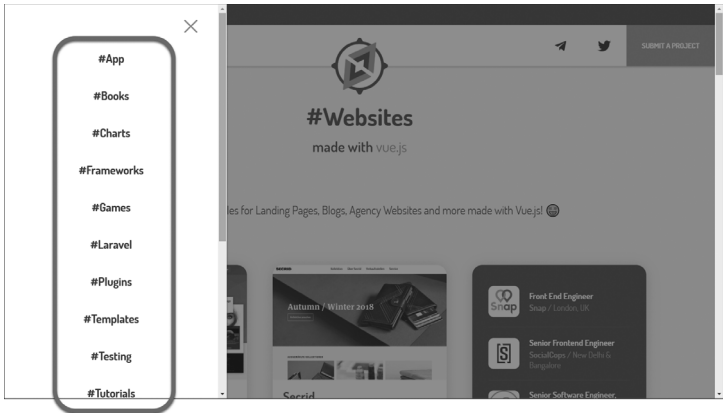
- **Nintendo** bekend van gameconsoles en allerlei Mario-games;
- **L'Oréal** bekend van talloze verzorgingsproducten, make-up, stylingadviezen en (veel) meer;
- **Alibaba** het grote Aziatische e-commercebedrijf;



Afbeelding 1.8 *In Nederland is bijvoorbeeld het bekende portaal mijnoverheid.nl gemaakt met Vue.js.*

- **Gitlab** een verzameling tools voor software development lifecycles;
- **Mijn Overheid** een portaal waar Nederlandse burgers communiceren met overheidsdiensten zoals belastingen, provincies, gemeenten en waterschappen.

Maar Vue wordt niet alleen ingezet bij dergelijke grote bedrijven. Er zijn talloze andere sites, variërend van onlineapplicaties en -boeken tot reisbureaus, contentmanagementsystemen, WordPress-templates en 'gewone' websites. Bekijk eventueel een overzicht op **madewithvuejs.com**.

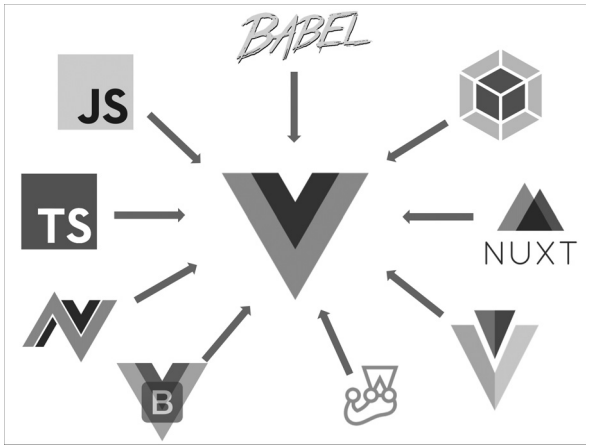


Afbeelding 1.9 *Op madewithvuejs.com kunt u meer voorbeelden vinden van sites en applicaties die met Vue zijn gemaakt. In sommige gevallen is ook de broncode beschikbaar.*

Het Vue.js-ecosysteem

Vue.js staat echter niet eenzaam in een woestijn, wachtend om gebruikt te worden. Er is een compleet ecosysteem ontwikkeld rondom Vue. We zagen al enkele aanvullende bibliotheken die in combinatie met Vue worden gebruikt, maar er is nog veel meer.

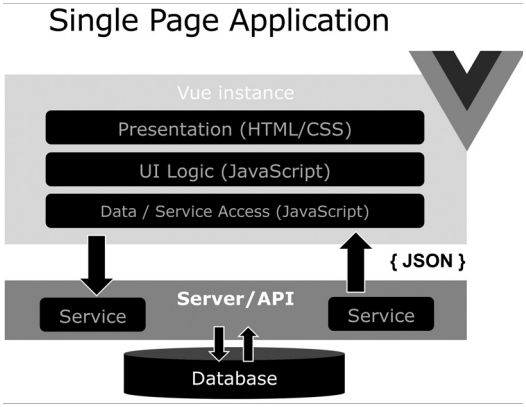
Er zijn aanvullende bibliotheken voor gebruikersinterfaces (bijvoorbeeld `vue-bootstrap` en `vuetify`), frameworks voor applicatieontwikkeling (bijvoorbeeld Nuxt) en aanvullende bibliotheken om mobiele applicaties voor iOS en Android mee maken (`vue-native`). Uiteraard is Node.js nodig als JavaScript-ontwikkelomgeving en nog veel meer. De afbeelding geeft een indruk van het ecosysteem dat in de loop der jaren is ontwikkeld. Maar ongetwijfeld komen hier nog veel meer onderdelen bij en is dit plaatje moeiteloos uit te breiden.



Afbeelding 1.10 *Er is een compleet ecosysteem rondom Vue ontwikkeld. Dit zijn aanvullende bibliotheken en technieken die goed samenwerken met Vue. Ze zijn niet verplicht (op JavaScript na), maar zorgen er vaak wel voor dat wensen eenvoudiger kunnen worden gerealiseerd.*

Architectuur van Vue-applicaties

In principe zijn Vue-applicaties toepassingen die volledig zelfstandig in de browser draaien. Er zijn ook varianten, zoals Vue in een standalone app die is gemaakt met Electron, Ionic of NativeScript. Daar gaan we in dit boek echter niet op in. We concentreren ons op webapplicaties met een architectuur zoals in afbeelding 1.11 te zien is.



Afbeelding 1.11 *De architectuur die we nastreven in Vue-applicaties: de logica- en presentatielaag draaien in de browser, datatoegang en authenticatie draaien op de server.*

Single Page Application

De architectuur die in afbeelding 1.11 wordt getoond, wordt ook wel het principe van *single page applications* genoemd. Dit betekent dat in één pagina (index.html) de logica van de applicatie geladen wordt en dat deze pagina steeds zichtbaar is in de browser. Vanuit index.html vindt alle navigatie plaats.

De app zelf bestaat natuurlijk uit veel meer dan één bestand. Elke component staat in zijn eigen bestand, er zijn CSS-bestanden, afbeeldingen enzovoort.

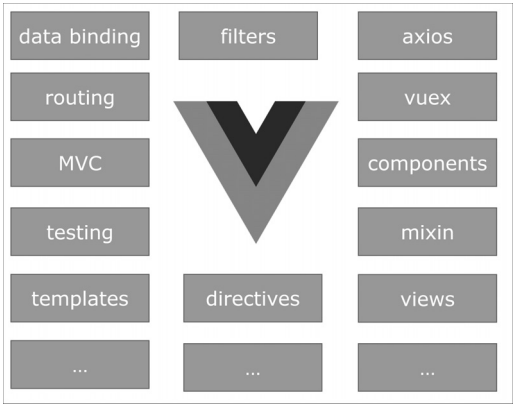
- De rol van de browser is:
  - *Presentatielaag* tonen aan de gebruiker, via de view van componenten. Dit is gewoon HTML en CSS zoals u kent, verrijkt met Vue-specifieke toevoegingen.
  - *Gebruikersinterfacelogica*, via het script van een view. Hierin staat logica om de gebruikersinterface samen te stellen, events te verwerken zoals muiskliks of het verwerken van Ajax-aanroepen en meer. Het scriptblok van een component wordt ook wel de controller genoemd.

    - *Data- en servicetoegang,* oftewel communiceren met een of meer RESTful API's op de server. De API praat vervolgens met de database en retourneert gegevens, het liefst in JSON-formaat. Datatoegang kan rechtstreeks vanuit een component, maar vaker wordt hiervoor een speciale service geschreven of wordt een *state management store* gebruikt. Hier gaan we later in het boek nog op in.
- De rol van de server is:
    - *Databasetoegang* via een API. Oneerbiedig gezegd wordt de rol van de server in moderne SPA-applicaties steeds verder teruggedrongen. Een server is niets meer dan een 'JSON-fabriek die data serveert'.
    - *Authenticatie*. Op het terrein van toegangsrechten is de server nog steeds onontbeerlijk. Immers, de complete Vue-app draait in de browser en is daarmee onveilig. Authenticatie en autorisatie zijn per definitie het domein van de server. Hier moet worden ingelogd met gebruikersnaam en wachtwoord, via sociale netwerken of anderszins. De server moet vervolgens een token of authenticatiecookie uitreiken (afhankelijk van de wijze van beveiliging die door de serverbeheerder is gekozen). De Vue-app is er verantwoordelijk voor dat dit token of de cookie met elk volgend verzoek wordt meegezonden.

In dit boek gaan we niet verder in op het inrichten of beheren van een webserver. In hoofdstuk 7 gaat u wel aan de slag met het communiceren met (bestaande) databases vanuit Vue.

Vue-begrippen

Om de architectuur uit afbeelding 1.11 te realiseren, moeten algemene termen als *presentation*, *ui logic* en *data/service-access* natuurlijk concreet worden gemaakt. Dit gebeurt in Vue-apps met begrippen als componenten, routing, mixins en meer. U ziet ze in afbeelding 1.12.



Afbeelding 1.12 *De algemene architectuur uit de vorige afbeelding wordt in Vue concreet gemaakt via dit soort begrippen.*

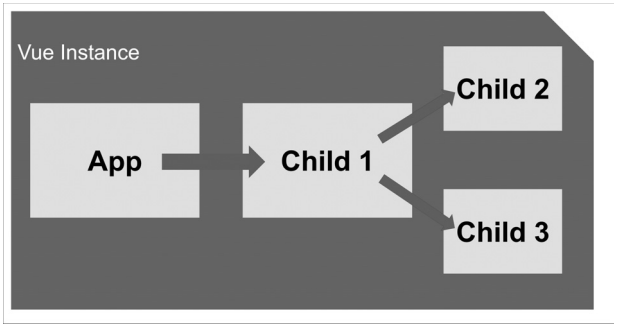
In de loop van dit boek zullen we steeds meer van de blokjes uit afbeelding 1.12 gaan invullen. We beginnen in hoofdstuk 2 met de begrippen componenten, templates en databinding. Daarna komen ook routing en andere onderdelen aan de orde.

De Vue-instantie

We hebben de term al een aantal keer genoemd. In de Vue-wereld staat het begrip *component* centraal. Componenten draaien echter niet zomaar vanzelf in de browser. Browsers zoals Firefox of Chrome weten niet vanzelf wat ze hiermee moeten doen. Componenten hebben vooralsnog een overkoepelend raamwerk nodig, waarmee ze met elkaar worden gekoppeld. Ook moet het beeld worden ververst met nieuwe componenten als de bezoeker daar om vraagt enzovoort. Dit overkoepelende raamwerk heet de Vue-instantie, of *Vue instance.*

Elke Vue-instantie heeft één hoofdcomponent. Dit heet de *root component*.

- Binnen die hoofdcomponent worden (meestal) deelcomponenten geladen.
- Elke deelcomponent heeft een eigen verantwoordelijkheid. Ze hebben een eigen gebruikersinterface en eigen logica.
- Deelcomponenten kunnen op hun beurt weer andere componenten bevatten. Deze worden met HTML-tags ingesloten in de template van de bovenliggende component.
- Het maken van een Vue-applicatie bestaat dus vaak uit het bouwen van meerdere kleine, zelfstandige componenten die met elkaar kunnen samenwerken. U bouwt aan een blokkendoos van componenten die onderling logische samenhang hebben en de app vormen.



Afbeelding 1.13 *Een Vue-app is een boomstructuur van componenten. Componenten draaien binnen een Vue-instantie.*

Een Vue-instantie wordt meestal gemaakt in het bestand main.js. Deze naam is niet verplicht, maar het is een goede afspraak om u hieraan te houden. Andere ontwikkelaars kunnen dan snel de structuur van uw app doorgronden. De inhoud van main.js is meestal eenvoudig en ziet er bijvoorbeeld zo uit als in het script.

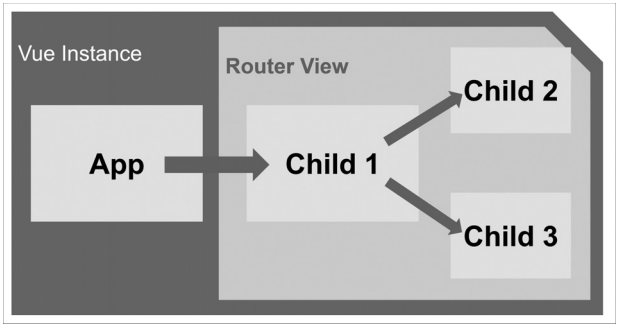
```
// main.js
import Vue from 'vue'
import App from './App.vue'

Vue.config.productionTip = false;
```

```
new Vue({
  render: h => h(App),
}).$mount('#app');
```

Vue-apps met routing

Complexere applicaties zullen ook gebruikmaken van de vue router. In dat geval wordt de structuur iets ingewikkelder. De Vue-instantie wordt dan uitgebreid met een routerinstantie. Alle componenten worden binnen de router getoond. In de code wordt een routerinstantie aangegeven met `<router-view>`. Hier gaan we vanaf hoofdstuk 6 dieper op in.



Afbeelding 1.14 *Een applicatie met routing. De componenten zijn zichtbaar binnen het door de router beheerde deel van de pagina.*

Wanneer naar nieuwe componenten wordt genavigeerd via het centrale hoofdmenu, een uitschuifbaar menu of gewone hyperlinks, wordt de URL in de adresregel van de browser bijgewerkt. Deze wordt dus ook door Vue beheerd.

De router wordt in dat geval toegevoegd aan de Vue-instantie tijdens het maken van de applicatie. Zo kan de Vue-instantie worden uitgebreid met andere onderdelen die globaal beschikbaar moeten zijn. U zult dit bijvoorbeeld ook zien in het hoofdstuk over state management met Vuex.