VUE framework

VUE framework

Gabriel Sánchez Cano

Boom beroepsonderwijs · Meppel · 2021

Boom beroepsonderwijs
info@boomberoepsonderwijs.nl
www.boomberoepsonderwijs.nl

Auteur: Gabriel Sánchez Cano
Redactie en opmaak: Henk Pel
Titel: Vue framework
ISBN 978 90 372 5752 6
Eerste druk / eerste oplage

Inhoud

Voorwoord

Opbouw

Bij de opbouw is gebruikgemaakt van de taxonomie van Romiszowski, waar onderscheid wordt gemaakt tussen kennis (het opslaan van informatie) en vaardigheden (acties uitvoeren om een doel te bereiken).
Elk lesblok eindigt met een kennistoets en een vaardigheid-lab. Elk hoofdstuk eindigt met een zelfstudieproject.

- Kennistoetsen: kennis van begrippen en procedures.
- Vaardigheid-labs: reproductieve vaardigheid, acties uitvoeren om een doel te bereiken.
- Zelfstudieprojecten: productieve vaardigheid.

In tegenstelling tot reproductieve vaardigheden doen productieve vaardigheden een beroep op de creativiteit en planningsvaardigheden van de student; ze gaan gepaard met (complexe) beslissingsvorming op bewust of onderbewust niveau. De student moet de geleerde informatie spontaan toepassen in nieuwe situaties, waarin niet van tevoren geoefend is. Er moeten nieuwe oplossingen voor nieuwe problemen bedacht worden.

Dit boek is met de grootst mogelijke zorg geschreven. De juistheid en volledigheid van de gegevens kunnen echter niet worden gegarandeerd. De auteur en uitgever aanvaarden geen aansprakelijkheid voor schade, van welke aard dan ook, die het directe of indirecte gevolg is van handelingen zoals onethisch hacken.

Ik wil al mijn studenten en collegae bedanken voor hun feedback tijdens het maken van dit boek. Speciale dank aan Bart Schrap voor zijn zorgvuldige feedback en commentaar. Ik heb dit boek met veel plezier geschreven en ik hoop dat zowel de studenten als docenten er met veel plezier mee zullen werken.

1 Inleiding Vue.js

De frontend developer heeft zich in een aantal jaar volledig ontwikkeld tot front-end engineer. Vroeger codeerde de frontend developer HTML met CSS en een beetje JavaScript. Tegenwordig wordt er veel meer verwacht van de frontend developer. Zij of hij krijgt onder andere met het volgende te maken:

- leren coderen met nieuwe frameworks;
- optimaliseren van code;
- de code compileren en transpileren;
- het managen van de *rendering* (weergave) van de code;
- schrijven van API's en andere databronnen;
- webserver configureren;
- de code inzetten;
- serverless apps coderen.

Enter de frameworks

Dit alles heeft geleid tot meer libraries, open source projecten en frameworks zoals **Vue**, **Angular** en **React**. Een software-framework is een herbruikbare software-omgeving die specifieke functionaliteit biedt voor het ontwikkelen en inzetten van software-applicaties. Van deze drie frameworks groeit **Vue** het snelst qua populariteit. **Vue** is zeer toegangkelijk en heeft de minst steile leercurve. Dit boek is een inleiding tot **Vue 3** en de belangrijkste modules en tools die het interessant en actueel maken.

Voorkennis

Kennis en ervaring met JavaScript, HTML en CSS, en het bouwen van webapplicaties is vereist.

Voor wie is dit boek?

Dit boek is voor software-ontwikkelaars die zich willen verdiepen in multi-platform reactieve en mobiele progressive web-applicaties (PWA). Progressive betekent dat de core library van Vue eenvoudig geïntegreerd kan worden met andere libraries of met bestaande projecten om een app-like ervaring te creëren. Een PWA is een applicatie die gebruikmaakt van nieuwe webtechnologieën die voorzien in een *app-like* ervaring.

Wat heb je nodig

Je hebt het volgende nodig om dit boek te kunnen gebruiken:

- Computer met internetverbinding
- Teksteditor/IDE
- Node.js
- Vue Cli
- Vue 3

1.1 Vue 3

Lesblok 1.1		
	Thema	Vue-architectuur
	Benodigdheden	Code-editor, Vue

In dit boek bespreken we de volgende definities en concepten om frontend development met Vue te doorgronden. Raapleeg deze lijst regelmatig.

Application state: is de state van de applicatie op een gegeven moment. De data in de application state zijn geïnitialiseerd bij het opstarten van de applicatie. Alle componenten van de applicatie hebben toegang tot de data. De data kunnen we muteren door middel van speciale events die plaatsvinden in de applicatie.

Content Distribution Network (CDN): zijn speciale, hoog performance servers die frameworks zoals Angular of Vue hosten met het doel om deze beter toegankelijker voor de gebruikers te maken. Vue wordt gehost in het volgende CDN: https://unpkg.com/vue.

Components: zijn de herbruikbare blokken van de applicatie met eigen data, functies en stijlen.

Declarative Views: deze views hebben een directe data binding (dataverbinding) tussen JavaScript-datamodellen en hun presentatie.

Directives: directives in Vue zijn instructies binnen HTML-elementen, bijvoorbeeld voor data binding.

Document Object Model (DOM): is een boom van nodes die de hiërarchische structuur van HTML-elementen in een document representeert.

Markdown: is een syntaxis voor het schrijven van teksten zonder markup (CSS en HTML-tags). Markdown-bestanden hebben een .md-extensie.

Model View ViewModel (MVVM): is een architectuurmodel waar de view en het datamodel met elkaar zijn verbonden. Vue is gebasseerd op het MVVM-architectuurmodel.

Model View Controller (MVC): is een architectuurmodel waar de controller de stroom van data tussen het datamodel en de view controleert.

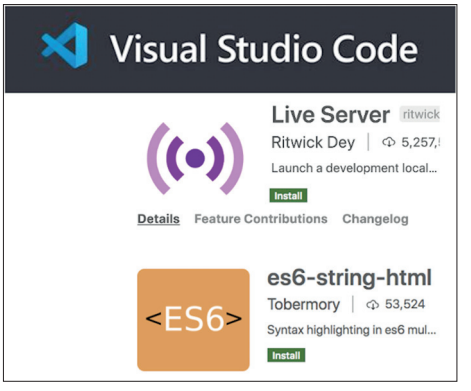
Node: is een open source server environment.

NPM: is een package manager in een Node voor het zoeken, installeren en managen van JavaScript packages.
One-way data binding: is eenrichting data binding waar wijzigingen in het datamodel automatisch worden weergegeven in de view.
Rapid prototyping: is een techniek voor het snel maken van mockups (proefmodellen) van gebruiker-interfaces.
Reactivity: is de reactie in de view op wijzigingen in het datamodel.
User interface (UI): zijn de visuele componenten die zorgen voor de interactie tussen de gebruikers en de applicatie.
Two-way data binding: is een tweerichting data binding waar wijzigingen in het datamodel automatisch worden weergegeven in de view en wijzigingen in de view automatisch worden opgeslagen in het datamodel.
Vuex: is de state-architectuur voor het managen van de application state.

Benodigde tools installeren

Om te kunnen beginnen met coderen in Vue 3 hebben we een aantal tools nodig.

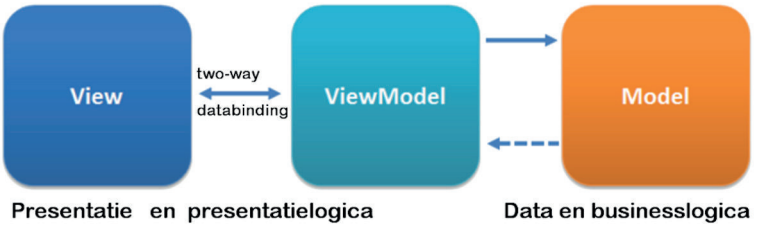
- *Opgave 1*
- Installeer Visual Studio Code of je favorite IDE.
- Installeer de Live Server extension in je editor om de resultaten van je scripts weer te geven.
- Installeer de ES6-string-html-extension in je editor om de ES6- en CSS-syntaxis met kleuren te markeren. In VSCode zien deze extensions er als volgt uit:



Figuur 1.1 Live Server- en ES6-string-html-extensies

Vue-architectuur

Vue (uitgesproken als *view*) is een progressive framework voor het bouwen van user interfaces. Vue is opgebouwd volgens de MVVM (*Model View ViewModel*) pattern architectuur.



Figuur 1.2 MVVM-componenten

Het MVVM-pattern

Het MVVM-pattern werd in 2005 geïntroduceerd door Microsoft-architecten Ken Cooper en Ted Peters met het doel om event-driven (gebeurtenis gestuurde) programmering van user interfaces te versimpelen. In de bovenstaande figuur zien we de componenten van het MVVM-pattern.

Een basis-app

De View

De *View* is de structuur en lay-out (presentatielogica) van het *Model* zoals gezien vanuit de gebruiker. De interactie tussen de gebruiker en de *View* loopt via het toetsenbord of de muis. Deze interactie wordt door het *ViewModel* behandeld door gebruik te maken van data binding.

- *Opgave 2: De View*

  Maak een nieuwe map met de naam **vuejs3** en open die met een editor, bijvoorbeeld VSCode. In deze map maak je een **index.html**-bestand aan met de volgende code:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Inleiding Vue 3</title>
    <!-- Import styles -->
    <link rel="stylesheet" href="./assets/my.css" />
    <!-- Import Vuejs 3 -->
    <script src="https://unpkg.com/vue@next">
    </script>
  </head>
  <body>
    <!-- Maak een <div> voor je app -->
    <div id="app">
      <!-- Data item {{pattern}} weergeven -->
      <h1>Pattern: {{ pattern }}</h1>
    </div>
```

```
    // Import je app vanuit main.js (ViewModel)
    <script src="./main.js"></script>
    <script>
    <!-- Je app (ViewModel) in id="app" invoegen -->
      const mountApp = app.mount("#app");
    </script>
  </body>
</html>
```

Om met het Vue-framework te kunnen werken hebben we de link naar het CDN (Content Distribution Network) voor Vue als volgt gecodeerd:

```
<script src="https://unpkg.com/vue@next">
```

Het element `<div id="app">` is de *View* en het presenteert de data, in dit geval het data-item met de naam pattern uit het *ViewModel*.

Het ViewModel

Het *ViewModel* is een abstractie van de *View* (properties en commando's) en heeft een *Binder* die de communicatie tussen de *View* en het *Model* automatiseert. Het *ViewModel* zit tussen de *View* en het *Model* in. Het volgende is een voorbeeld van een *ViewModel*.

- *Opgave 3: Het ViewModel*

In de map **vuejs3** maak je het volgende **main.js**-bestand met de volgende code:

```
// De app creëren
const app = Vue.createApp({
  data() { },
  methods: { },
  computed: { },
});
```

Hier hebben we het *ViewModel* als **app** gemaakt. In de app kunnen we onder andere een data-object (dit is het *Model*) en een methods-object coderen voor eigen methodes die op de data opereren en een computed object voor methodes voor caching. Deze methodes bespreken we in de volgende hoofdstukken.

Het Model

Het *Model* is de data access layer (businesslogica) en de data store. Het volgende is een voorbeeld van het *Model* in Vue.

- *Opgave 4: Het Model*
  Zorg ervoor dat in de map **vuejs3** het data-object in **main.js** er als volgt uitziet:

```
// De app creëren
const app = Vue.createApp({
  data() {
    return {
      // data-item creëren
      pattern: "MVVM",
    };
  },
  methods: { },
  computed: { },
});
```

Het *ViewModel* serveert de data (*Model*) aan de *View.*

Expressies

Met accolade-expressies **{{ }}** kunnen we JavaScript binnen HTML coderen. De expressie **{{ pattern }}** is de expressie voor het weer te geven (extrapoleren) data-item in het *Model*. Voorbeelden van expressies zijn:

```
{{voornaam + ' ' + achternaam}}
```

en

```
{{clicked ? true : false}}
```

Het MVVM-pattern probeert de voordelen van het MVC-pattern (logische scheiding van functionaliteit) en de voordelen van data bindings te benutten. Zie het boek *Objectgeörienteerd programmeren* van Boom beroepsonderwijs.

Met de rechtermuisknop klik je op je **index.html**-code in je editor en vervolgens op *Open with Live Server*. Als resultaat zie je de data van het *Model* weergegeven in de *View*. Dit alles wordt uitgewerkt in het *ViewModel*.

Output

```
Pattern: MVVM
```

Vue 3 Reactivity

Reactivity is het systeem in Vue 3 waarbij de *View* reageert op veranderingen in het *Model*. Wanneer er de waarde van het data-iten **pattern** in het *Model* verandert is de nieuwe waarde in de *View* weergegeven.

Kennistoets 1.1: Inleiding Vue 3

Bestudeer de theorie van dit lesblok en maak de volgende kennistoets.

Toets 1.1	
	1. Wat is VUE?
	2. Wat is een software framework?
	3. Wat is MVVM?
	4. Wat doet de View?
	5. Wat is het Model?
	6. Wat doet het ViewModel?

Vaardigheid-lab 1.1: Inleiding Vue 3

Voeg een nieuw data-item met de naam *bedrijfsnaam* met de waarde FietsShop in je data (*Model*) toe. Zorg ervoor dat dit data-item in de *View* wordt weergegeven. Het resultaat moet er als volgt uitzien:

Output

```
FietsShop
```

1.2 Vue-directives

Lesblok 1.2		
	Thema	Vue-directives
	Benodigdheden	Code-editor, Vue

Vue-directives zijn instructies in HTML-elementen. Zoiets als HTM-attributen in templates. Alle Vue-directives (commando's) beginnen met de `v-`prefix. In de volgende opgaven kijken we naar de volgende Vue-directives:

- v-if
- v-bind
- v-text
- v-html

Conditional binding met v-if

Met de conditional `v-if`-directive kunnen we beslissingen uitvoeren. Dit doen we binnen HTML-elementen. De conditional Vue-directive coderen we als volgt:

- *Opgave 5*
  Open **index.html** en voeg de volgende code binnen `<div id="app">`

```
<p v-if="uur < 12">Goedemorgen!</p>
<p v-else-if="uur >= 12 && uur < 18">Goedemiddag!</p>
<p v-else-if="uur >= 18">Goedenavond!</p>
<p v-else >Waarde in uur ongeldig!</p>
```

  Open **main.js** en voeg het volgende `data-item` toe:

```
uur: new Date().getHours(),
```

  #### Output
  Als resultaat zie je in je browser de juiste groet.

Attribuut binding met v-bind

De `v-bind`-directive synchroniseert de data in je app met je HTML-attributen in je View door gebruik te maken van **declarative data binding**-technologie. Dit doen we in de volgende opgave.

- *Opgave 6*
  Update **index.html** en voeg onderaan het volgende div-element toe.

```
<div class="fiets">
  <div class="image">
    <img v-bind:src="fietsimage" />
  </div>
</div>
```

  Open **main.js** en voeg het volgende `data-item` toe:

```
fietsimage: "assets/batavus.png"
```

  In je map `vue3js` maak je de nieuwe map `assets`. Zorg ervoor dat je een afbeelding voor **batavus.png** in de map `assets` toevoegt.

  Hier hebben we een div voor een fiets en een div voor de fietsafbeelding gecodeerd. Met `v-bind` hebben we de waarde van het HTML--attribuut met het data-item `fietsimage` in **main.js** verbonden. Bijvoorbeeld `<script src="./main.js">` verwijst naar een brondocument.Attribuut-binding kunnen we ook bij de volgende HTML-attributen coderen:

- alt
- href
- title
- class
- id

De verkorte form van `v-bind:` is simpelweg het `:`-teken.

```
<img v-bind:src="fietsimage" />
```

is hetzelfde als

```
<img :src="fietsimage" />
```

- *Opgave 7*
  Codeer in de map **assets** de volgende stijlen en sla dit op als **my.css**.

```
img,
  .info {
    width: 300px;
    text-align: center;
  }
```

Output

Als resultaat zien we de volgende weergave:

Figuur 1.3 Data binding

v-text

We kunnen de directive `v-text` gebruiken in plaats van extrapoleren met accolades **{{ }}**, bijvoorbeeld:

```
<p v-text="info"></p>
```

is hetzelfde als

```
<p>{{info}}</p>
```

- *Opgave 8*
  Open **index.html** en voeg binnen de fiets-div de volgende HTML-paragraaf toe:

```
<p class="info" v-text="fietsinfo"></p>
```

Open **main.js** en voeg het volgende **data**-item toe:

```
fietsinfo:
`Een ideale allrounder.
Een perfect stabiel frame.
Met de Batavus haal je een echte winnaar in huis!`
```

Als we in JavaScript een lange tekst opdelen in meerdere regels krijgen we een foutmelding. Om een lange tekst van meerdere regels te coderen gebruiken we het accent grave (`).

Output

Als resultaat wordt de fietsinfo weergegeven.

v-html

We kunnen de directive **v-html** gebruiken om teksten met HTML-tags weer te geven.

- *Opgave 9*

Update **index.html** zodat de fietsinfo-HTML-paragraaf er als volgt uitziet:

```
<p class="info" v-html="fietsinfo"></p>
```

Update **main.js** en zorg dat het data-item **fietsinfo** er als volgt uitziet:

```
fietsinfo:
`<h4>Een ideale allrounder</h4>
Een perfect stabiel frame.
Met de Batavus haal je een echte winnaar in huis!`
```

Output

Deze HTML-tekst wordt met de directive **v-html** in **index.html** als volgt weergeven:

Een ideale allrounder
Een perfect stabiel frame. Met de **Batavus** haal je een echte winnaar in huis!