Mentor course - Advanced JavaScript
Vue.js
Andreas Baggesgaard
Cphbusiness Academy
Maj, 2017

Kom i gang med Vue.js

"The progressive JavaScript Framework"

Indledning

Denne guide har fokus på at gøre det nemmere at komme i gang med Vue.js. Ofte tager det tid at sætte sig ind i et nyt framework, og dokumentationen kan indeholde utrolig meget information, der har det med at gøre en mere forvirret fra start af. Derfor vil jeg prøve at give mit bud på, hvad man bør fokusere på, og dermed hurtigst muligt komme i gang med at udnytte mange af Vue.js' fordele.

Min tilgang har været at få information fra youtube, forums og selve dokumentationen. Jeg vil anbefale at gennemgå denne youtube serie på 10 afsnit, der virkelig klæder dig godt på:
https://www.youtube.com/watch?v=vzSjlLzGB1A&list=PLwAKR305CRO_1yAao-8aZiQnBqJeyng4O

Al koden til denne guide kan findes her: https://github.com/andreasbaggesgaard/vuejsexamguide

Vue.js dokumentationen: https://vuejs.org/v2/guide/

Tilgang

Vue.js dokumentationen er egentlig rimelig overskuelig, og "Introduction" sektionen giver et godt indblik i hvordan det fungerer. Jeg vil derfor prøve at fokusere mere på opsætningen af et Vue.js projekt, og demonstrere hvor kraftfuld frameworket er.

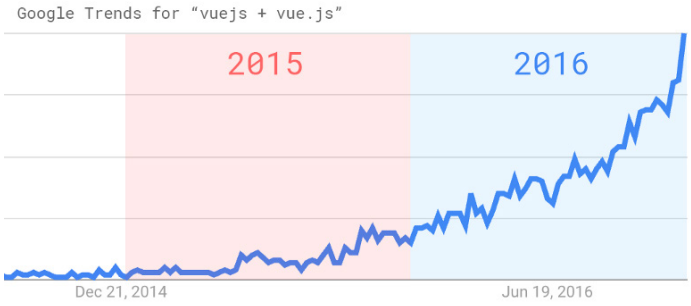
Derfor anbefaler jeg:
1. Udforsk dokumentationen.
2. Se de 10 afsnit på youtube (de varer cirka 5-10 minutter per afsnit).
3. Følg min guide

Som dokumentationen også kommer ind på, så kræver det kendskab til HTML, CSS og JavaScript ellers kan det godt være op af bakke. Men bare en generel forståelse for data objekter og programmering, så hvis du ikke har den store erfaring med JavaScript, men et andet sprog, så burde det stadigvæk være rimelig lige til.

Hvorfor Vue.js?

Webudvikling i dag, bliver domineret utrolig meget af JavaScript, da det kan rigtig meget, og både google og facebook har lavet deres egne populære frameworks, Angular og React, der har været med til at gøre JavaScript populært igen. Da de to frameworks, har to af de største spillere bag sig, og dermed er i hurtig udvikling, så kan de også opfattes som mere komplicerede, og læringskurven kan være stejl.

Det er der Vue.js kommer ind i billedet. Vue.js er skabt af Evan You, der har arbejdet hos google, men nu er fuldtid på dette projekt. At der kun er én udvikler på det, er egentlig ret imponerende, men det kan også være en svaghed, hvis udviklingen af frameworket kommer til at gå for langsomt. Ind til videre står det i hvert fald ikke stille, og Vue.js' popularitet er steget gevaldigt det seneste års tid, siden den officielle lancering i 2016.



Vue.js er ikke revolutinerende på nogen måde i forhold til nytænkning, men det har prøvet at simplificere tingene, og har taget det bedste fra andre frameworks. Fx. bruger det en "Virtual DOM" som React, og er komponent baseret ligesom fx Angular. Der er lavet forskellige tests, og Vue.js er faktisk hurtigst på mange måder, hvilket virkelig også kan mærkes.

Du kan læse mere om sammenligningen af frameworks her: https://vuejs.org/v2/guide/comparison.html

Det fede ved Vue.js er virkelig hvor hurtigt du kan komme i gang med det. Jeg har rodet lidt med React og Angular, og opsætningen kan tage en evighed før du har et projekt kørende. Jeg har ikke den store erfaring med at kode i dem, så en fair sammenligning kan jeg ikke give. Jeg har dog sammenlignet koden i et Angluar 2 projekt med et af mine Vue.js projekter, og det ligner utrolig meget hinanden i form af opsætning og kodestruktur, så efter du har lært Vue.js, så vil du helt klart nemmere kunne skifte til et andet framework, hvis det er.

Vue.js kommer af ordet "View", så der er ikke nogen "server-side rendering", og er lavet primært til at bygge advancerede interfaces (frontend), der er koblet op på en eller anden API. Frameworket har dog en masse "contributors", der har samarbejdet med Evan You om at lave plugins, der fx kan tilbyde mere advanceret "routing", og også "server-side rendering". Så selvom det kan virke mere simpelt end andre frameworks, så er der en stor opbakning til projektet, og det er desuden et af de mest populære "repositories" på GitHub for tiden.

Så hvis man gerne vil kunne et framework, der giver lige så gode jobmuligheder som React og Angular, så tror jeg at Vue.js i den nærmeste fremtid kunne blive en del af eliten. Ifølge Evan You, så er et af de næste skridt også at kigge på "Native rendering", så der også kan laves IOS/Andriod apps, så det er en spændende fremtid vi går i møde.

Jeg har i hvert fald ikke fortrudt, at jeg kastede mig ud i det her framework, da jeg virkelig er blevet grebet af det. Jeg plejede at lave min frontend kode i jQuery, og ofte ender du med en masse linjer kode til selv de mere simple ting. Så at kunne få en mere struktureret tilgang til det, med en virkelig kraftig motor, der er ligeså nemt at installere som jQuery, har været fantastisk.

Når man starter ud med at kode i Vue.js, så kan man bare tilføje et link til Vue.js i sin HTML fil, og så er du kørende. Men som sagt, så vil jeg fokusere mere på det mere kraftfulde ved frameworket. Derfor vil jeg guide dig igennem opsætningen af et Vue.js projekt, og brug af "Single File Components". Mit mål er at vise hvordan man bør tænke som udvikler, når man skal strukturere sin kode, og gerne have "DRY" (don't repeat yourself) principperne i baghovedet.

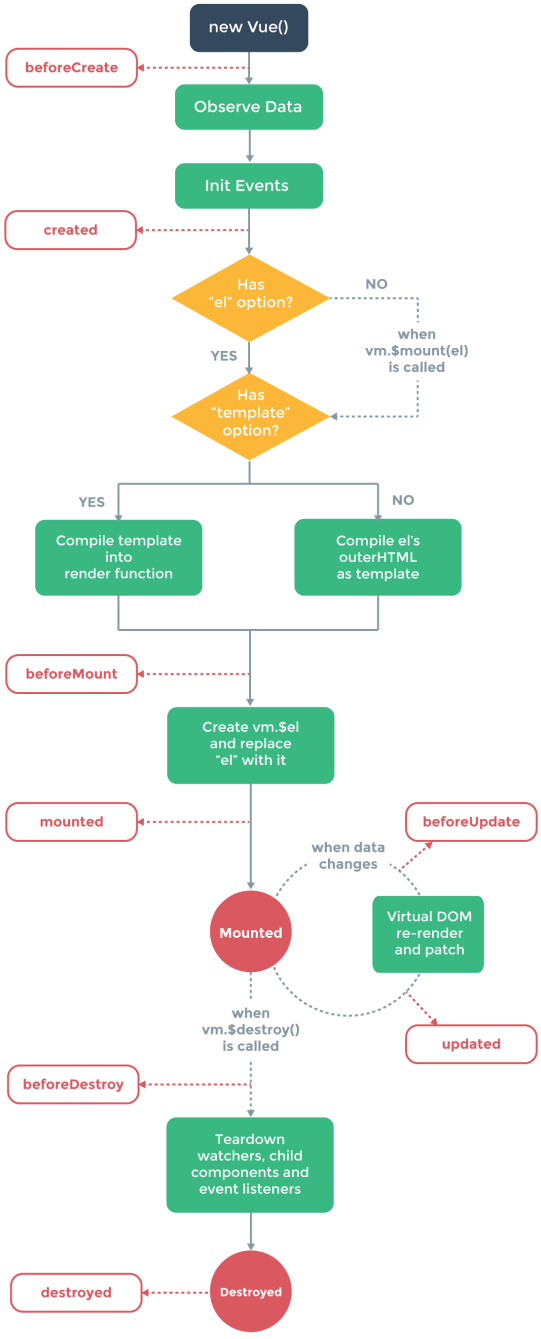
Inden vi går i gang, så bør vi lige kigge på en vigtig ting for at kunne forstå opbygningen af "Vue Objects".

```
var vm = new Vue({
  data: {
    a: 1
  },
  created: function () {
    // `this` points to the vm instance
    console.log('a is: ' + this.a)
  },
  watch: {},
  mounted: {},
  updated: {},
  destroyed: {},
  methods: {},
})
```

Et "vue object" bliver deklaret på følgende måde, som ses på billedet. Det er bygget op omkring MVVM princippet, altså data i mellem view'et og modellen.

Dernæst har objektet forskellige "stages", som det gennemgår når det bliver instantieret. Det har fx. "created", der foregår i starten, og "mounted" der foregår nær slutningen. Så alt efter hvordan din logik er, så kan du tilføje din kode til de forskellige "stages".

Du kommer måske fra en MVC baggrund, og de her "stages" fungerer i princippet som vue objektets "controllers".

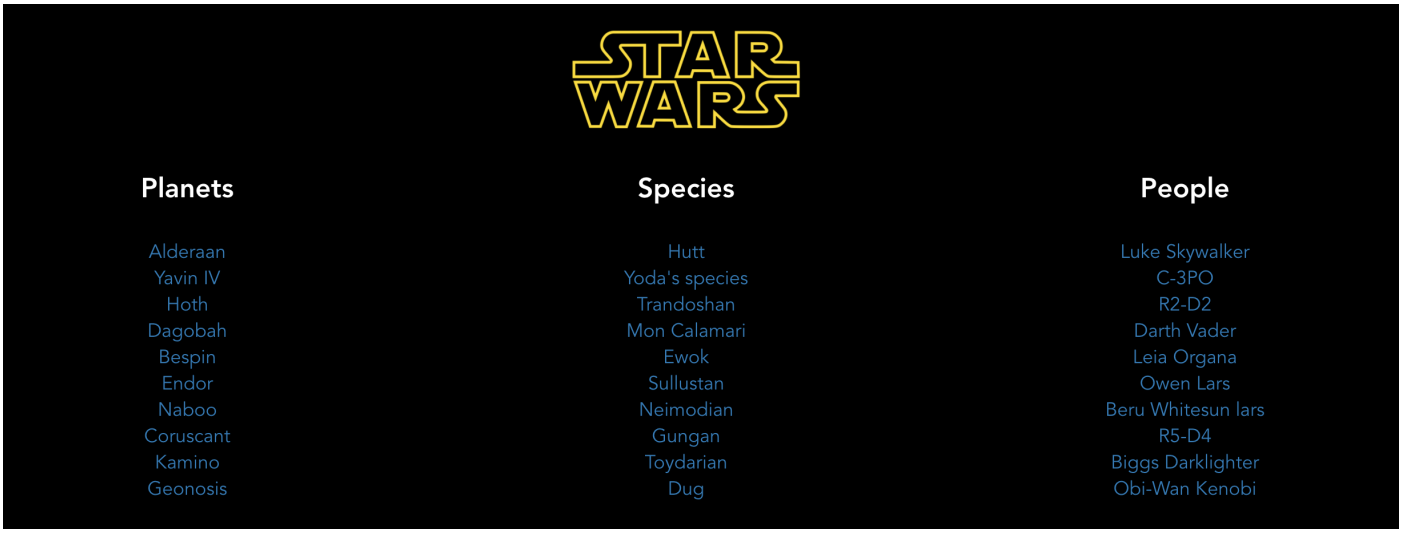


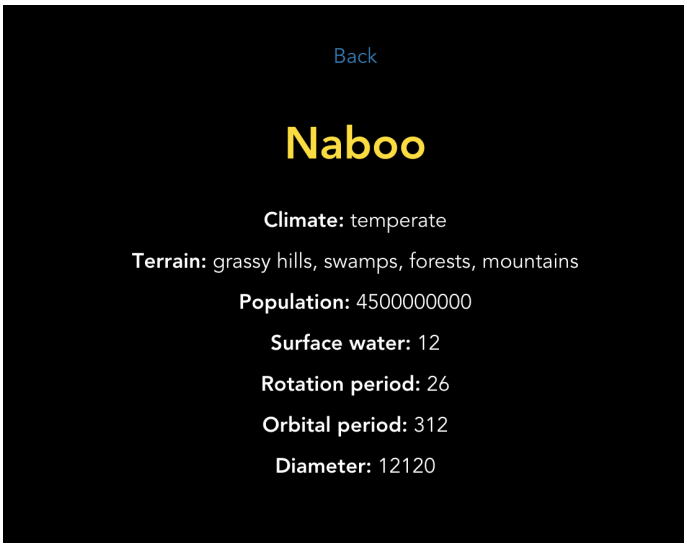
Lad os komme i gang!

1. Opsætning af projekt

2. Brug af "Single File Components"

For at gøre det så simpelt og forståeligt som muligt, så vil vi lave en lille applikation, der benytter en Star Wars API, kaldet SWAPI, der gør det muligt at tilgå JSON data omkring hele Star Wars universet. Idéen er at få en liste over planeter, arter og personer, og derefter kunne trykke på dem for at få mere information.





Tanken med dette er at lave en "reuseable" liste komponent, man både kan bruge på forsiden, og på selve information siden. På den måde undgår vi at skrive næsten den samme kode om og om igen, og kan bare indsætte komponenten, der hvor der er brug for den.

1. Opsætning af projekt

Hvis du ikke allerede har gjort det, så skal du først installere Node.js, så du kan benytte NPM (package manager). Du kan benytte dette link: https://nodejs.org/en/.

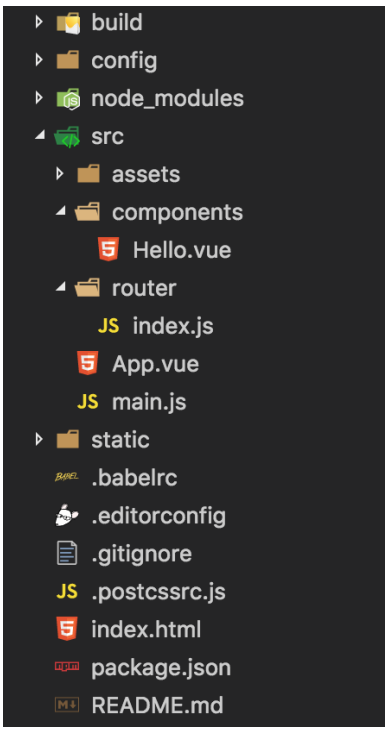
Næste skridt er at åbne din terminal, og gennemgå følgende skridt:

```
# install vue-cli
$ npm install --global vue-cli
# create a new project using the "webpack" template
$ vue init webpack my-project
# install dependencies and go!
$ cd my-project
$ npm install
$ npm run dev
```

Du vil undervejs blive spurgt om at installere forskellige plugins til Vue.js, men du behøver kun at sige ja til "vue-router".

```
[? Project name test
[? Project description test
[? Author ab
 ? Vue build standalone
[? Install vue-router? Yes
[? Use ESLint to lint your code? No
[? Setup unit tests with Karma + Mocha? No
[? Setup e2e tests with Nightwatch? No
```

Når du er færdig, så har du nu installeret et helt nyt Vue.js projekt, der åbner i dit browser vindue når du skriver "npm run dev". Det fede er at du får et top moderne arbejdsmiljø, der benytter "webpack" og "browserify", der så at sige er med til "compile" dine vue filer til bl.a. .js - og .css filer.



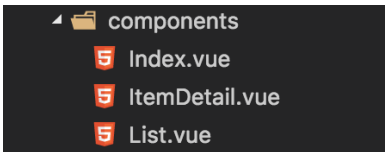
Dit setup ser nu ud som på billedet. De vigtigste ting du skal forholde dig til er:

- **Components** (Der hvor dine "Single File Components" er placeret).
- **Router, index.js** (Opsætning af "Single Page Application" routing).
- **App.vue** (Det sted hvor din Vue applikation bliver instantieret).
- **Main.js** (Import af plugins til vue objektet).
- **index.html** (standard HTML setup).

For at komme ind på hvad en "Single File Component" er, så indeholder den et template tag, der kan bruges til at blande normal HTML med Vue JavaScript. Dernæst et script tag, der indeholder "export default", der er en del af "ES6 module system", der gør det muligt eksportere din komponent til en anden fil via import. Til sidst er der et style tag, der selvfølgelig indeholder CSS.

Så på den måde, er det en meget intuitiv tilgang med HTML, JavaScript og CSS i samme fil.

2. Brug af "Single File Components"



I min lille Star Wars applikation, så har jeg lavet 3 komponenter:

- En "**Index**" fil til at skabe forsiden.
- En "**ItemDetail**" fil til mere information siden.
- En "**List**" fil til selve liste funktionaliteten.

Index.vue

```
<template>
  <div class="index">

    <div class="row">

      <img src="../assets/logo02.png"  width="200"/><br /><br />

      <div class="col-md-4">
        <h3>Planets</h3><br />
        <list datasource="http://swapi.co/api/planets/"></list>
      </div>

      <div class="col-md-4">
        <h3>Species</h3><br />
        <list datasource="http://swapi.co/api/species/"></list>
      </div>

      <div class="col-md-4">
        <h3>People</h3><br />
        <list datasource="http://swapi.co/api/people/"></list>
      </div>

    </div>

  </div>
</template>

<script>
import list from '@/components/List'

export default {
  name: 'index',
   components: {
    'list': list,
  }
}
</script>

<style>
```

Som det kan ses, så har Index komponenten importeret liste komponenten. Der er brugt bootstrap til at lave et layout, og selve listen er kodet sådan, at den kan blive tildelt en "datasource", som den henter data fra. Der foregår ikke meget JavaScript her, men når vi nu skal til at kigge på "List.vue", så sker der lidt mere.

List.vue

```
<template>
  <div class="data-list">

      <div v-for="item in starwarsdata">
          <div v-if="item.name == selectedItem">

              <h1>{{item.name}}</h1><br />

              <div class="loading" v-if="loading">
                  <h2>Loading...</h2>
              </div>

              <!-- People -->
              <p v-if="item.birth_year"><b>Gender:</b> {{item.gender}}</p>
              <p v-if="item.height"><b>Height:</b> {{item.height}}</p>
              <p v-if="item.birth_year"><b>Birth year:</b> {{item.birth_year}}</p>
              <p v-if="item.skin_color"><b>Skin color:</b> {{item.skin_color}}</p>
              <p v-if="item.hair_color"><b>Hair color:</b> {{item.hair_color}}</p>
              <p v-if="item.eye_color"><b>Eye color:</b> {{item.eye_color}}</p>

              <!-- Species -->
              <p v-if="item.language"><b>Language:</b> {{item.language}}</p>
              <p v-if="item.classification"><b>Classification:</b> {{item.classification}}</p>
              <p v-if="item.average_height"><b>Average height:</b> {{item.average_height}}</p>
              <p v-if="item.skin_colors"><b>Skin color(s):</b> {{item.skin_colors}}</p>
              <p v-if="item.hair_colors"><b>Hair color(s):</b> {{item.hair_colors}}</p>
              <p v-if="item.eye_colors"><b>Eye color(s):</b> {{item.eye_colors}}</p>
              <p v-if="item.average_lifespan"><b>Average lifespan:</b> {{item.average_lifespan}}</p>

              <!-- Planets -->
              <p v-if="item.climate"><b>Climate:</b> {{item.climate}}</p>
              <p v-if="item.terrain"><b>Terrain:</b> {{item.terrain}}</p>
              <p v-if="item.population"><b>Population:</b> {{item.population}}</p>
              <p v-if="item.surface_water"><b>Surface water:</b> {{item.surface_water}}</p>
              <p v-if="item.rotation_period"><b>Rotation period:</b> {{item.rotation_period}}</p>
              <p v-if="item.orbital_period"><b>Orbital period:</b> {{item.orbital_period}}</p>
              <p v-if="item.diameter"><b>Diameter:</b> {{item.diameter}}</p>
          </div>

          <div v-if="showAllItems">
              <router-link :to="{ name: 'item', params: { id: item.name }}">{{item.name}}</router-link>
          </div>

      </div>


  </div>
</template>
```

Her ses template opbygningen i ren HTML. Der er brugt "reactive directives", der kan forbindes med vue objekt instansen. Her er der fx blevet brugt "v-for" og "v-if", der fungerer som et "foreach loop" og en "if statement". Til sidst bliver der brugt "vue-router" til at kunne lave et link med en parameter, der kan sendes med til "itemDetail". Vi kigger nærmere på opsætningen af routing senere.

Fortsættes på næste side -->

```
<script>
export default {
  props:['datasource'],
  name: 'data-list',
  data () {
    return {
      starwarsdata: [],
      selectedItem: '',
      showAllItems: true,
      loading: false,
    }
  },
    created: function(){
        this.retrieveData();
        this.getItemDetails();
    },
    methods: {
        retrieveData: function(){
            this.loading = true;
            let xhr = new XMLHttpRequest();
            let self = this;
            xhr.open('GET', this.datasource);
            xhr.setRequestHeader('Content-Type', 'application/json');
            xhr.onload = function(){
                let data = JSON.parse(xhr.responseText);
                self.starwarsdata = data.results;
                self.loading = false;
            }
            xhr.send();
        },
        getItemDetails: function(){
            let url = window.location.href;
            let getParameter = "";
            let parameterValue = "";

            try {
                getParameter = decodeURIComponent(url);
                parameterValue = getParameter.substr(getParameter.indexOf("detail/") + 7)

                if (document.location.href.indexOf("detail/") > -1) {
                    console.log(parameterValue)
                    this.selectedItem = parameterValue;
                    this.showAllItems = false;
                }
            } catch (e) {
                console.error(e);
            }
        },
}
</script>

<!-- Add "scoped" attribute to limit CSS to this component only -->
<style scoped>

</style>
```

Her er så JavaScript'en til liste komponenten. Der er skabt 2 funktioner, hvor den ene henter data fra "datasource", som vi har set specificeret tidligere i Index.vue. Den sidste funktion kigger på url'en om hvorvidt der er en parameter, og hvis der er, så er det kun data'en der passer til parameteren, som bliver vist.

Så derfor kan liste komponenten blive brugt flere steder, og vise forskellige listevisninger. Det smarte er helt klart også muligheden for at kunne give en "datasource", da man kan genbruge komponenten mange gange, og give den hvilken som helst url. Navnet "datasource" afhænger forresten af hvad du kalder det i "props", som du kan se øverst. På den måde kan du sende data til komponenten.

ItemDetail.vue

```
<template>
  <div class="item-detail">

    <router-link to="/" class="link">Back</router-link><br /><br />

    <list datasource="http://swapi.co/api/planets/"></list>
    <list datasource="http://swapi.co/api/species/"></list>
    <list datasource="http://swapi.co/api/people/"></list>

  </div>
</template>

<script>
import list from '@/components/List'

export default {
  name: 'item-detail',
  data () {
    return {}
  },
    components: {
    'list': list,
  }
}
</script>

<!-- Add "scoped" attribute to limit CSS to this component only -->
<style scoped>

</style>
```

Den sidste komponent indeholder bare de 3 lister. Lige nu giver det måske ikke så meget mening hvorfor vi har brug for den, men på næste side viser jeg hvordan "routing" er sat op, og der har vi brug for den.

index.js

```
import Vue from 'vue'
import Router from 'vue-router'
import ItemDetail from '@/components/ItemDetail'
import Index from '@/components/Index'

Vue.use(Router)

export default new Router({
  routes: [
    {path: '/', component: Index},
    {path: '/detail/:id', name: 'item', component: ItemDetail }
  ]
})
```

"Routing" fungerer således at man kan sende komponenter afsted, alt efter hvilken "side" man er på. Da det er en "Single Page Application" vi bygger, så foregår der ikke "page load", og alt er "ajax".

Som man kan se på billedet, så specificerer man en sti/path, og hvilken komponent der skal benyttes dertil. Det er også muligt at benytte "parameters", der indeholder et "id".

Du kan derefter bruge det i dit "router-link", og give den det data, som du føler for.

```
<router-link :to="{ name: 'item', params: { id: item.name }}">{{item.name}}</router-link>
```

Nu har vi været det hele igennem, og du har set hvor kraftfuld frameworket er, og fordelen ved at bruge komponenter. Som sagt, så kan du finde al koden på github, og har mulighed for at få projektet op at køre.

Hvad nu ?

Hvis du nu har været igennem mine 3 anbefalede punkter, så er du godt klædt på til at benytte Vue.js. Næste skridt kunne være at kigge på firebase, der er perfekt til at bygge web apps. Det kommer lidt an på hvad du bygger og hvilke behov du har, men firebase er en database API du nemt kan benytte, og kan dermed bygge applikationer uden et "server-side" sprog.

Ellers udforsk Vue.js dokumentationen endnu mere, og slip fantasien løs. Vue.js gør det nemmere at bygge mere komplicerede applikationer, som ellers ville tage en evighed at bygge i fx jQuery eller normal JavaScript.