Kennismaken met Svelte

Nu in coronatijden mijn ZZP-agenda maar heel mager is gevuld, heb ik besloten om van de nood een deugd te maken. Ik ga twee nieuwe frameworks onderzoeken waarvoor het mij tot op dit moment aan de tijd ontbrak om me er meer in te verdiepen. Dat zijn:

- ✓ Svelte, op https://svelte.dev/
- ✓ Flutter, op https://flutter.dev/

In dit artikel kijk ik naar Svelte (spreek uit 'Svelt'). Afgelopen week heb ik kennisgemaakt met Svelte, de documentatie gelezen, online tutorial(s) gevolgd en verder rondgesnuffeld naar Svelte-informatie op de voor de hand liggende plekken als Stack Overflow, Medium en Dev.to.

Disclaimer - ik ben frontend developer. Ik ken frameworks als Angular, Vue en React op mijn duimpje. Hiermee heb ik veel applicaties gemaakt. Mijn beleving en beschrijving van Svelte komt dan ook vanuit dit perspectief. Ik maak regelmatig de vergelijking me zaken die ik ken. Dat hoeft voor jou natuurlijk niet zo te zijn! Toch probeer ik het zo eenvoudig mogelijk te houden en een voor iedereen duidelijke beginners/intro te schrijven over Svelte.

In dit artikel beantwoord ik de volgende vragen:

1. Wat is Svelte?
2. Waar wordt het voor gebruikt?
3. Hoe gaat lokale Installatie en Ontwikkeling?
4. Hoe maak ik een eerste applicatie?
5. Hoe maak ik een niet-triviale *proof-of-concept* applicatie: een app die op basis van een zoekvraag van een gebruiker communiceert met een backend en de resultaten in de UI toont.

Omdat ik een absolute beginner ben in Svelte heb ik ongetwijfeld zaken gemist of heb ik dingen omslachtig gedaan (omdat ik die nu eenmaal ken uit Angular, React of Vue) die in Svelte veel eenvoudiger kunnen. Aarzel in dat geval niet om het mij te laten weten!

1. Wat is Svelte?

Svelte is een framework voor het ontwikkelen van webapplicaties. Het is daarmee vergelijkbaar met Angular, React en Vue. Het is *niet* vergelijkbaar met de vorige generatie webdevelopment tools zoals jQuery, AngularJS of Backbone. Svelte is een modern framework, gebouwd op Node.js, ES6, componenten en JavaScript-bundlers. Svelte is een *alternatief* voor Angular, React of Vue. Of zo je wilt: een concurrent. Je gebruikt het een of het ander. Niet tegelijk.

Svelte is gemaakt door Rich Harris, de ontwikkelaar die ook de bundler `rollup.js` heeft gemaakt. Het is dan ook geen verrassing dat rollup de standaard bundler is in Svelte. Je kunt eventueel ook WebPack instellen als bundler, maar dat moet je zelf doen.

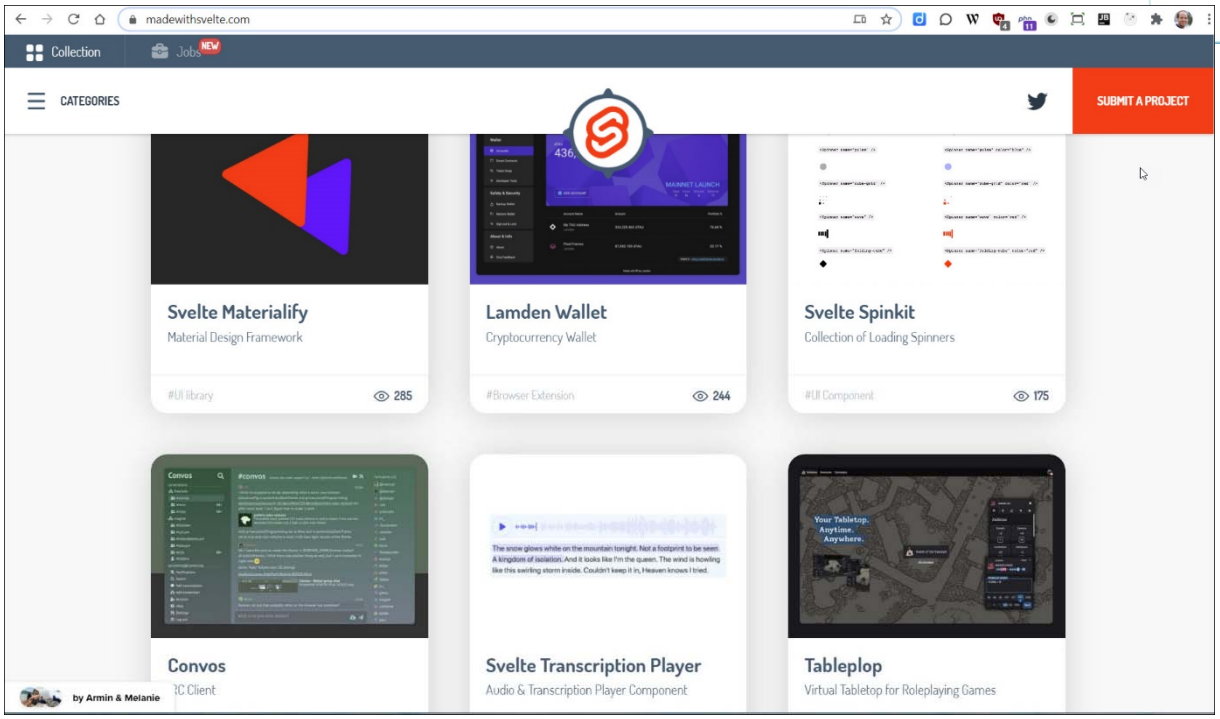
Het belangrijkste verschil tussen Angular, React en Vue is dat Svelte-applicaties in hun geheel *op de server* worden gecompileerd. Alles wordt vertaald naar JavaScript en kant-en-klaar naar de browser gestuurd. Svelte is daarom razendsnel. Bedenk, in andere frameworks wordt ook het framework zélf meegestuurd naar de browser. De browser moet vervolgens de code parsen en uitvoeren. Die stap wordt bij Svelte overgeslagen. Svelte kent geen *virtual dom* of *shadow dom*.

Lees er meer over op https://svelte.dev/blog/virtual-dom-is-pure-overhead.

2. Waar wordt Svelte voor gebruikt?

Svelte wordt gebruikt voor het ontwikkelen van grotere webapplicaties. Natuurlijk kun je er ook een website voor 'de bakker op de hoek' mee maken, maar daar is het framework in principe niet voor bedoeld. Denk voor typische toepassingen aan:

- E-commerceapplicaties
- Dashboards
- CRUD-Frontends voor databaseapplicaties
- Een overzicht van websites en -applicaties die zijn gemaakt met Svelte is beschikbaar op https://madewithsvelte.com/.
-



3. Hoe gaat lokale ontwikkeling en installatie?

Svelte heeft naar goed gebruik een CLI-omgeving (net zoals Angular, React en Vue). Deze wordt echter niet lokaal geïnstalleerd. Rich Harris heeft een kleine tool geschreven (`degit`) die een Github-repository ophaalt *zonder* de .git-info. Daardoor is ook deze tool razendsnel. Je geeft aan degit een sjabloonlocatie mee en op basis daarvan wordt een lokaal project gemaakt. Dit stel je in met `npm install`, waarna je vervolgens de eenvoudige startapplicatie (`Hello World`) kunt draaien.

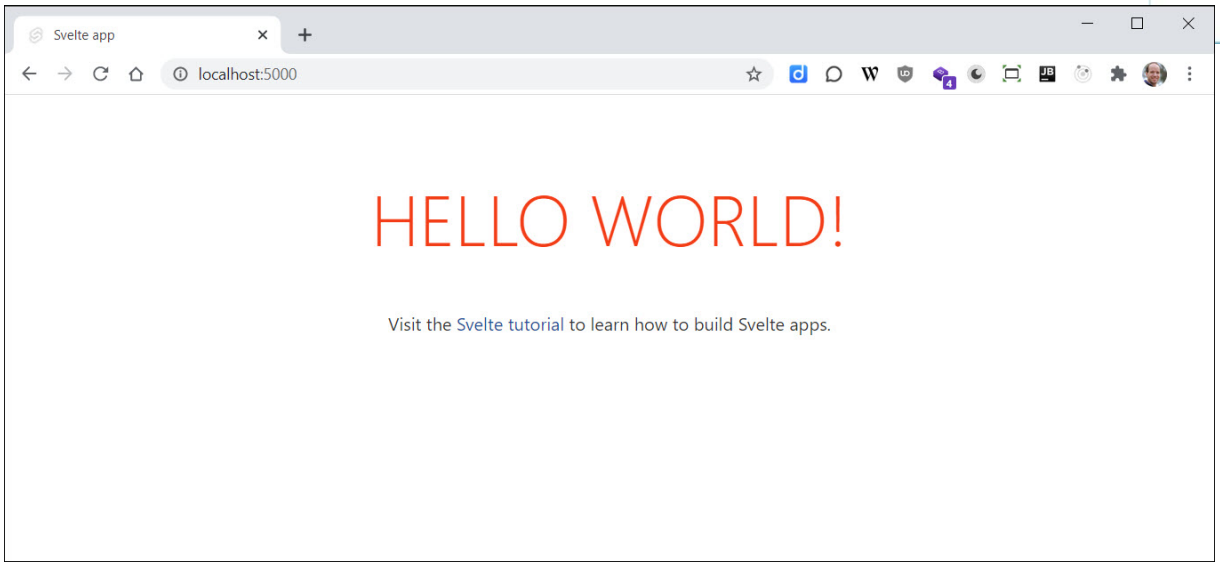
De opdrachten om een Svelte-project te maken zien er daarmee uit als:

```
npx degit sveltejs/template my-svelte-project
cd my-svelte-project

npm install
npm run dev
```

Je applicatie wordt gestart op http://localhost:5000.

Daarna kun je het project `svelte-hello-world` openen in je favoriete editor (Visual Studio Code of JetBrains WebStorm) en er verder in werken.

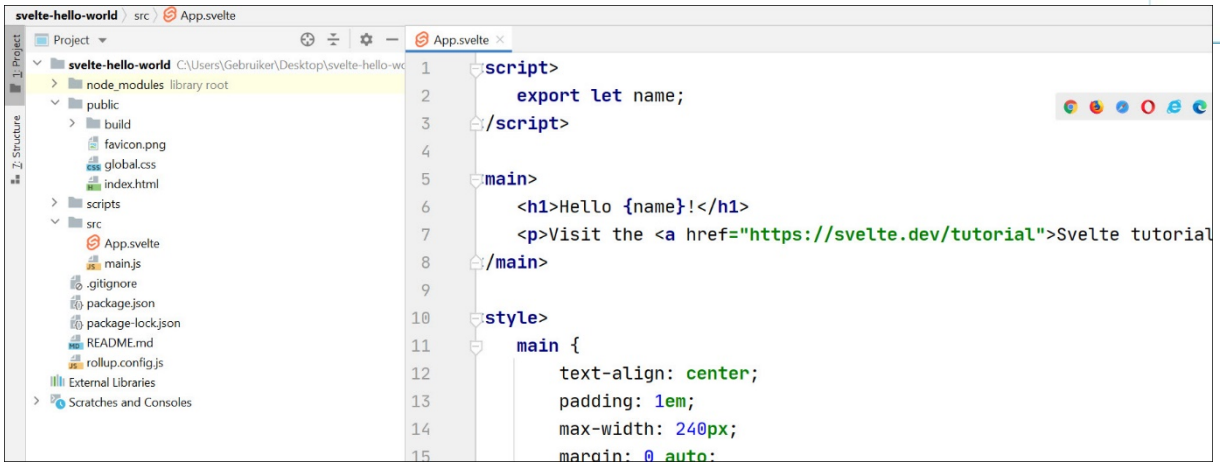


4. Hoe maak ik een eerste applicatie?

Nadat je de applicatie hebt geïnitialiseerd en gestart op de hiervoor beschreven wijze, kun je componenten toevoegen of bestaande componenten aanpassen. Net als in de andere frameworks zijn *componenten* de basis bouwstenen van een applicatie. Je maakt dus geen webpagina's meer, maar componenten die tezamen de lay-out van de pagina vormen.

De basisapplicatie ziet er uit als in de volgende afbeelding. Wanneer je eerder hebt gewerkt met frameworks als Angular, Vue of React ziet dit er vertrouwd uit. Heb je hier nog geen kennis van, dan is de structuur niet erg lastig.

- ✓ In de map `\public` staat je `index.html` en plaats je andere statische bestanden (css, afbeeldingen, pdf's enzovoort).
- ✓ In de map `\src` staan je componenten en overige logica. De applicatie wordt gestart via `main.js` (typisch zul je hier nooit iets in wijzigen), de startcomponent is `App.svelte`.
- ✓ Op het rootniveau (`\`) staan configuratiebestanden als `package.json`, `readme.md` en `rollup.config.js`. Ook hier zul je maar heel zelden handmatig wijzigingen in hoeven aan te brengen.
- ✓ Het meeste zul je aan het werk zijn in de map `\src`. Maak hierin een bijvoorbeeld een map `\components` en verdeel dit eventueel verder onder in modules, afhankelijk van de requirements van je project.



Plug-ins voor Svelte-ontwikkeling

Componenten worden in svelte opgeslagen als `.svelte`-bestanden. Deze worden door de meeste editors standaard niet herkend. Toch wil je zaken als kleurcodering, automatisch aanvullen en meer. Hiervoor hebben de meeste IDE's gelukkig plug-ins of extensions beschikbaar. Installeer bijvoorbeeld:

- In Visual Studio Code: de extension `Svelte for VS Code`, op https://marketplace.visualstudio.com/items?itemName=svelte.svelte-vscode.
- In WebStorm: de plug-in `Svelte` door Tomasz Blachut, op https://plugins.jetbrains.com/plugin/12375-svelte.
- Voor andere editors (Atom, Sublime, Eclipse) zul je er naar moeten zoeken, of `.svelte`-bestanden aanmelden als JavaScript- of `.html`-bestanden. Dan krijg je tenminste kleurcodering.

Componenten in Svelte

Componenten bestaan in Svelte uit drie delen.

- ✓ Een `<script>`-blok waarin alle logica van de component staat.
- ✓ Een HTML-blok waarin de template/user interface van de component staat.
- ✓ Een `<style>`-blok waarin eventuele CSS-stijlen van de component worden genoemd. Stijlen hebben standaard *component scope* (net zoals in Angular en Vue). Svelte kent geen *CSS-as-JavaScript*, zoals React.

Basisstructuur van een .svelte-component

```
<script>
        // Alle script op de pagina
</script>

<main>
<!--    HTML/template van de pagina. Hier een een <main></main> tag, -->
<!--    maar dit is niet verplicht.     -->
</main>

<style>
```

Alle onderdelen zijn optioneel en de volgorde maakt niet uit. Je mag dus ook een component maken die uit een leeg bestand bestaat (!). Niet dat dit nuttig is, maar het toont de flexibiliteit van Svelte. Het framework gaat uit van *Single File Components*, net zoals React en Vue (en anders dan Angular). Alle logica en lay-out wordt dus ingekapseld in een enkel bestand. Er is geen *separation of concerns by file types*. In de component importeer je eventueel andere componenten die je vervolgens als HTML-element kunt aanspreken. Bijvoorbeeld op deze wijze:

Svelte syntaxis

Componenten schrijf je in een typische Svelte-syntaxis. Het is feitelijk een mix van Angular en Vue aan de ene kant, omdat je veel HTML-schrijft die wordt uitgebreid met Svelte-specifieke directives. Aan de andere kant lijkt het ook op React, omdat je in Svelte-componenten rechtstreeks JavaScript-expressies tussen enkele accolades mag schrijven. Zoals in `<h1>Hello {name}</h1>`. De syntaxis is daarmee tamelijk beknopt (er zijn handige shortcuts), maar wel behoorlijk wennen.

Daarbij zijn er veel Svelte-specifieke uitbreidingen zoals

`{#if …} … {/if}`

`{#each …} … {/each}` en

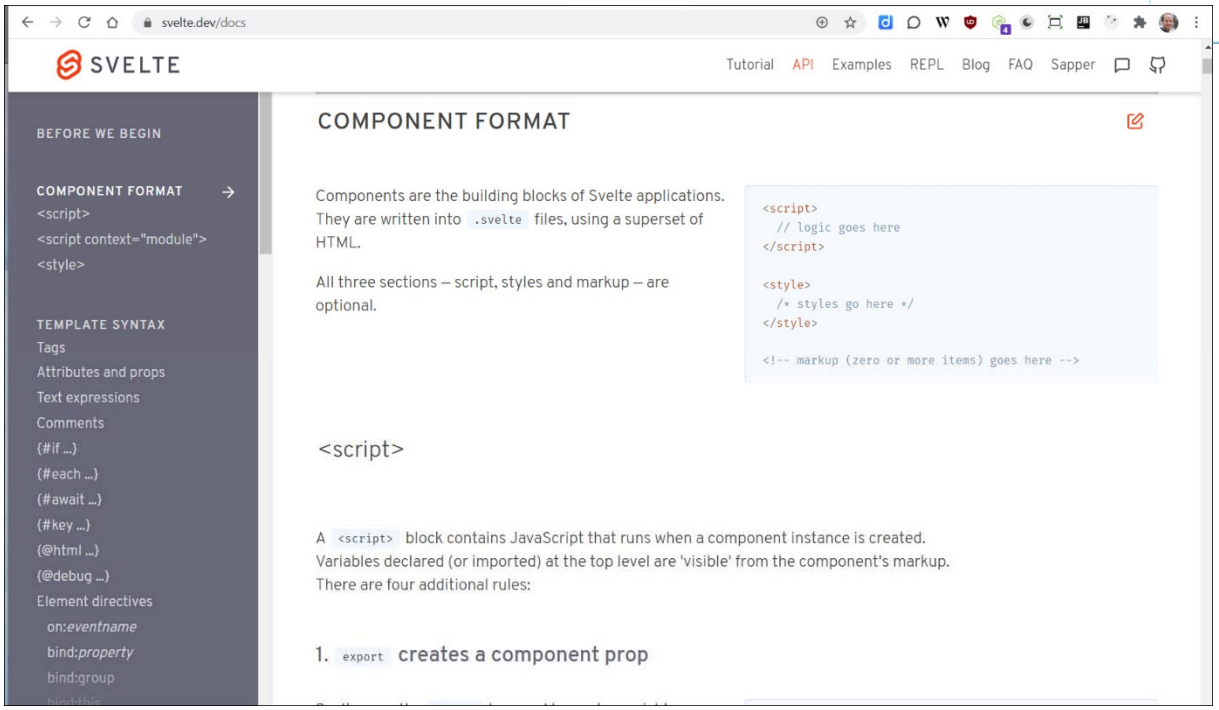
`{#await …}`

Dit zorgt naar mijn mening soms voor rommelige opmaak van de HTML-code. Het is zeker helderder en duidelijker dan React JSX, maar bepaald niet zo compact als `*ngFor` van Angular of `v-for` van Vue. Ik wéét waarom het gedaan is, maar het is…wennen. Laten we het daar maar op houden.

Svelte kent vergelijkbare bindings als Vue en Angular. Dit zijn onder meer

- ✓ **Simple data binding** met `{…}` zoals je al zag
- ✓ **Event binding** met `on:`, zoals in `on:click={<someHandler>}` of `on:mouseover={…}`.
- ✓ **Attribuut binding** waarbij je een variabele rechtstreeks kunt binden aan de waarde van een attribuut zoals in `<img src={src}>`. Als de naam en de waarde van het attribuut aan elkaar gelijk zijn (zoals hier), mag je dit ook afkorten als `<img {src}>`. Supercompact en erg elegant. Properties mag je daarnaast met de spread-operator aan een element toekennen, zoals in `<MyElement {…props}/>`. Je hoeft ze dan niet apart te benoemen. Mooi gedaan.
- ✓ **Two-way binding** met `bind:`. Je kunt bijvoorbeeld een variabele toekennen aan een tekstinvoerveld en de waarde hiervan direct uitlezen door `<input type="text" bind:value={<someVariable>}>` te gebruiken. De directive `bind:` is daarnaast voor tal van andere zaken te gebruiken. Ook dit gaat erg intuïtief als je er aan gewend bent. Hier toont Svelte zich een beetje een combinatie van Vue en React naar mijn idee.

Alle overige mogelijkheden en syntaxis wordt uitstekend beschreven in de Svelte documentatie op https://svelte.dev/docs. Dit is maar weer eens het bewijs dat de kwaliteit en gebruikersvriendelijkheid van een framework vaak valt of staat met de documentatie die er voor beschikbaar is. Die van Svelte is uitstekend in orde. Ik heb maar zelden een beroep hoeven doen op Stack Overflow of andere bronnen om Svelte te leren.



Tutorial

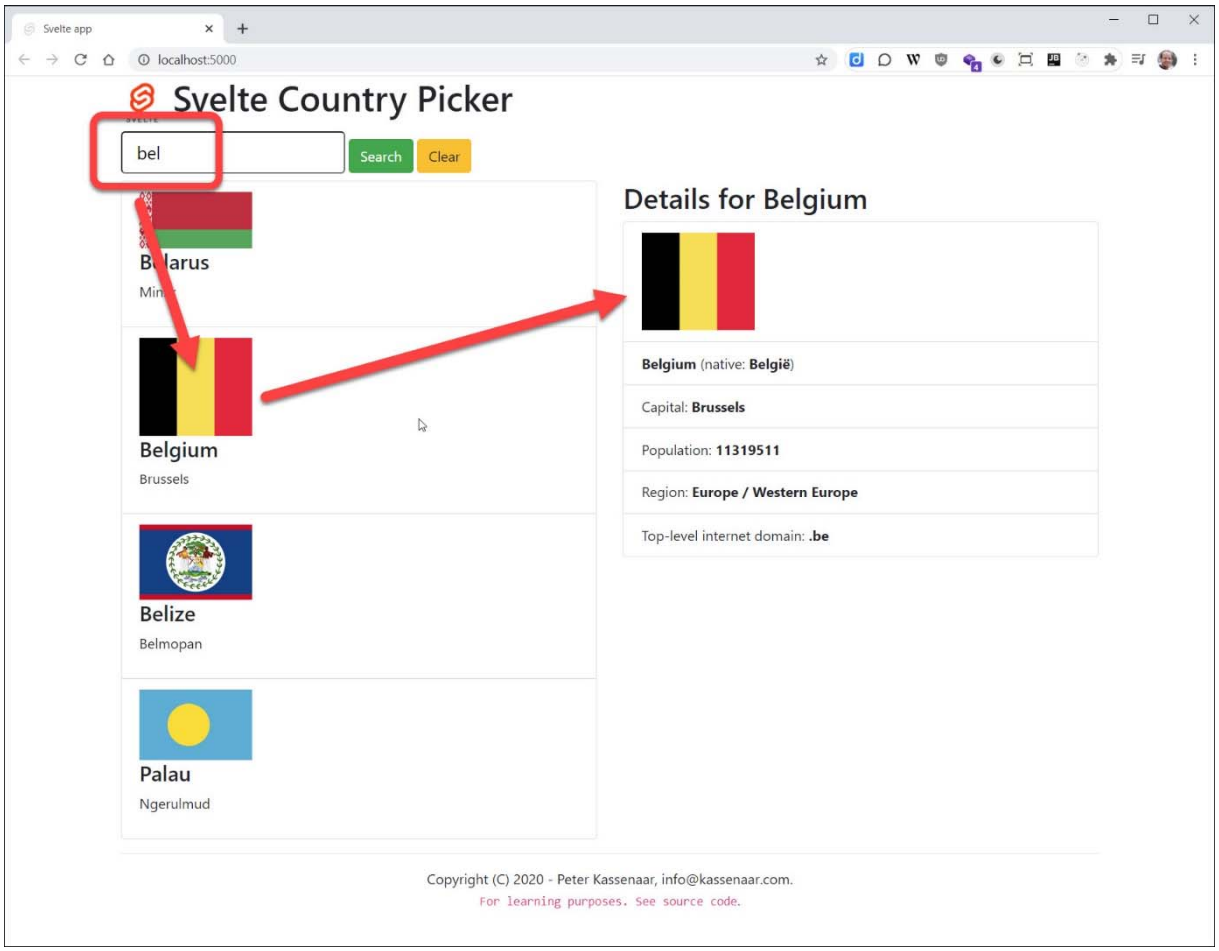
Bij het leren van Svelte heb ik erg veel gehad aan de interactieve tutorial, beschikbaar op https://svelte.dev/tutorial/basics. De schatting van de makers dat je "in ongeveer anderhalf uur wel door de tutorial heen bent" is evenwel een schromelijke overschatting. Reken op ruim een dag, als je alles goed wilt uitproberen en zo nu en dan een zijstap wilt maken. Of ik ben niet zo snel. Dat kan ook natuurlijk.

5. Hoe maak ik een Proof-of-concept applicatie?

Met alle kennis uit de tutorial was ik na een paar uurtjes echter al in staat om een applicatie als onderstaande te maken. De requirements zijn tamelijk eenvoudig, maar laten toch het complete stappenplan van een real-life applicatie, inclusief communicatie met een backend zien.

- ✓ Toon een eenvoudige user interface waar de bezoeker kan zoeken op (deel van) de naam van een land.
- ✓ Communiceer met een backend - ik gebruik hiervoor de REST Countries API, beschikbaar op https://restcountries.eu/ - om een serie landen op te halen die voldoen aan het trefwoord van de bezoeker.
- ✓ Toon een lijstje met landen die aan het trefwoord voldoen in de user interface.

- ✓ Bij klikken op een land worden details van het betreffende land getoond in een nieuwe component.



Je kunt deze applicatie zelf bekijken op github, op https://github.com/PeterKassenaar/svelte-demo-countries. Voel je vrij om de applicatie te downloaden, er een `npm install` voor uit te voeren en te kijken hoe de data flow door de applicatie en de componenten is. Vervolgens kun je hem verder zelf uitbreiden. Mocht je aanpassingen of verbeteringen hebben, dan houd ik me natuurlijk aanbevolen voor PR's!

Dit is een tamelijk eenvoudige applicatie, nog zonder error handling, routing of een state management-oplossing. Dit zou echter wel relatief eenvoudig toe te voegen zijn. Svelte heeft hiervoor mogelijkheden aan boord, maar met het oog op de tijd en het leerproces heb ik die niet toegepast. Dat zou de code een stuk complexer maken dan nodig is.

Conclusie

Svelte is een framework dat zich qua functionaliteit zeker kan meten met de meer bekende oplossingen als Angular, Vue en React. Het heeft superveel potentie en brengt eigenlijk het beste uit alle frameworks samen in een nieuwe omgeving. Doordat de browser geen overhead heeft maar kant-en-klare JavaScriptcode krijgt, is Svelte in potentie het snelste van alle frameworks verwacht ik. Hierbij moet ik wel aantekenen dat ik geen audit heb gedaan alle varianten heb vergeleken qua snelheid, footprint en andere performance-metrics.

De state management-oplossing die Svelte standaard aan boord heeft (met *stores*) behoort zelfs tot de beste die ik ooit heb gezien. Zoveel eenvoudiger dan Angular met Ngrx en React met Redux, en zelfs beter schaalbaar (denk ik) dan in Vue met Vuex. Dit is wat state management in de andere frameworks eigenlijk had moeten zijn!

Met *Sapper* (https://sapper.svelte.dev/) is er bovendien een applicatieframework beschikbaar dat draait 'bovenop' Svelte. Sapper is bedoeld voor het maken van applicaties en lijkt daarmee op Nuxt (Vue) of Next (React).

Toch is Svelte naar mijn mening nog niet klaar voor het 'grote werk'.

Dit komt niet door de mogelijkheden van het framework zelf, maar waarschijnlijk puur door de leeftijd ervan. Svelte is nog jong en onvolwassen. Ik miste bijvoorbeeld:

- ✓ Een aangewezen oplossing voor **routing**. Onontbeerlijk voor grotere webapplicaties. Er zijn wel tal van 3rd-party mogelijkheden waar door de makers naar verwezen wordt, maar een aanbevolen, officieel ondersteunde oplossing is naar mijn idee noodzaak.
- ✓ Nog weinig **user interface-libraries**. Het is geen probleem om standaard Bootstrap of een andere library te implementeren en er zijn bijvoorbeeld Svelte-implementaties van Bootstrap (*SvelteStrap*) en Material Design (*Svelte Material UI*), maar het is allemaal nog erg mager.
- ✓ **Testing**. Evenmin als voor routing is er momenteel een goede unit testing-oplossing beschikbaar. Je kunt zelf aan de slag met Cypress, maar er is geen officiële ondersteuning voor. Dat komt natuurlijk mede doordat Svelte al op de

server wordt gecompileerd en er geen framework in de browser draait. Daardoor zijn frameworks als Jasmine, Jest of Karma niet bruikbaar. Hopelijk wordt hier in de toekomst aan gewerkt.

- ✓ Beperkte ondersteuning voor **TypeScript**. TypeScript is een technologie die steeds bredere ondersteuning krijgt in het frontend. Hoewel er standaard een script setupTypeScript.js aanwezig is in nieuwe projecten, kent Svelte geen out-of-the-box ondersteuning voor TypeScript. Er zijn flink wat scherpe randjes en je moet er nog heel wat moeite voor doen. Ook bij Vue en React is TypeScript een *opt-in* keuze, maar je ziet dat deze frameworks al jaren langer de tijd hebben gehad om zich hierop aan te passen. Ook dit zal - hopelijk - een kwestie van tijd zijn.

Al met al denk ik genoeg redenen om Svelte zeker uit te proberen en te bekijken of het een oplossing is in jouw situatie. Mijn indruk is dat voor serieuze enterprise-toepassingen nog iets te vroeg is. Maar houd Svelte zeker op je radar voor toekomstige projecten. En natuurlijk laat ik mij graag terechtwijzen! Kom maar op met je projecten waarin je het tegendeel bewijst.

Auteur bio

Dit artikel is geschreven door *Peter Kassenaar*. Peter heeft meer dan 20 jaar ervaring in webdevelopment en heeft talloze boeken over webdevelopment en verwante technieken gepubliceerd. Hij is een bekende trainer in West-Europa en reist de wereld over om training te geven. Peter staat bekend om zijn intensieve, geconcentreerde manier van lesgeven en praktische hands-on workshops. Daarbij verliest hij echter nooit het belang van de individuele deelnemer uit het oog. Hij biedt veel persoonlijke aandacht en ondersteuning tijdens de workshops en oefeningen. Je kunt Peter volgen op Twitter (@PeterKassenaar, Nederlands en Engels) en LinkedIn (www.linkedin.com/in/peterkassenaar/). Zijn persoonlijke website is te vinden op www.kassenaar.com.

Ik geef training in frontend technieken zoals Angular, React en Vue - en binnenkort ook Svelte. Ook in coronatijd, ook virtueel, ook op jouw bedrijf. Laat weten wat je leervraag is, dan zoeken we een oplossing!