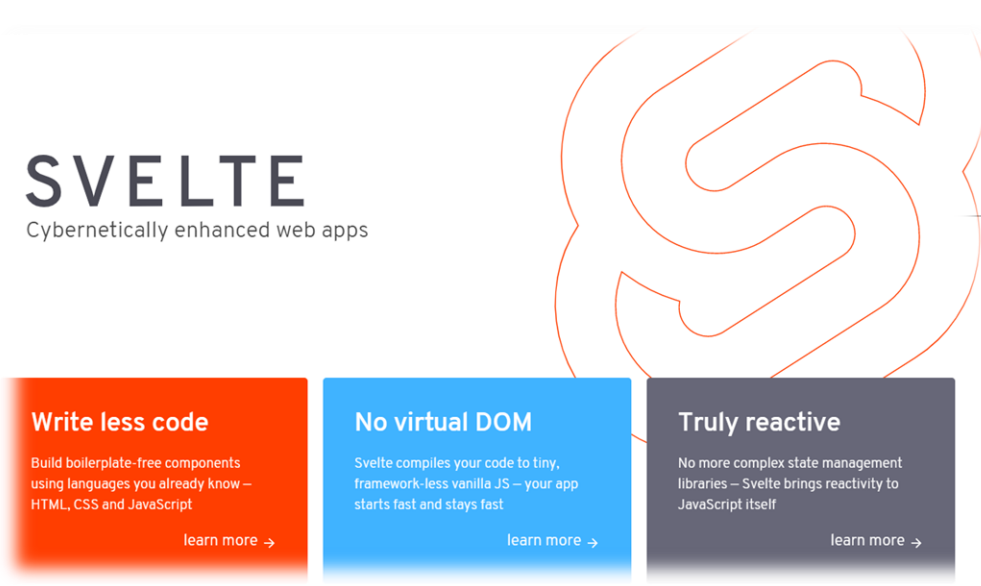
Intro to Svelte 3 & Sapper

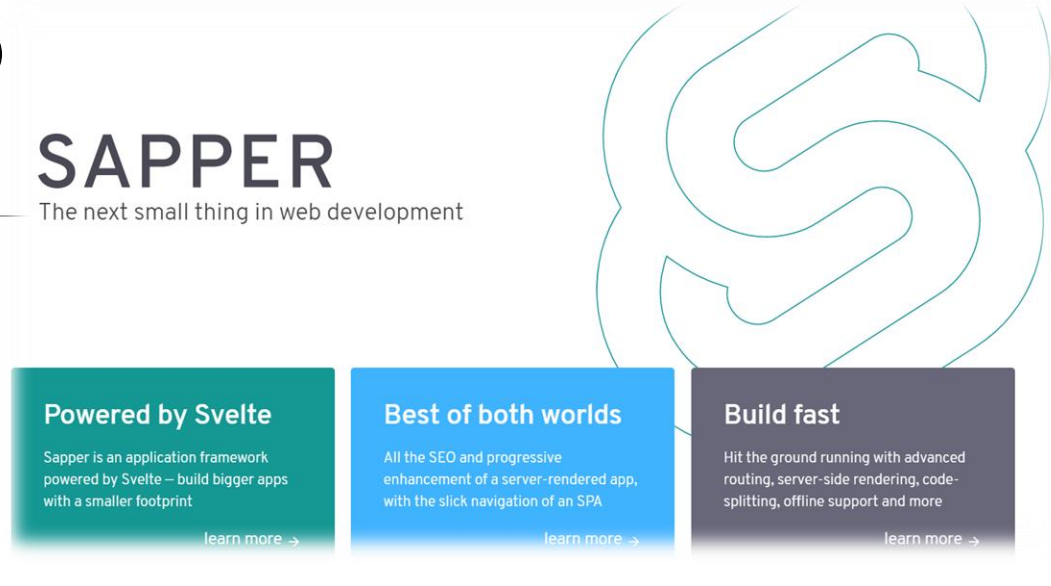
The return of 'Write Less, Do More'

By

Julio ZINGA

(02/21/2020)





Svelte & Sapper-intro

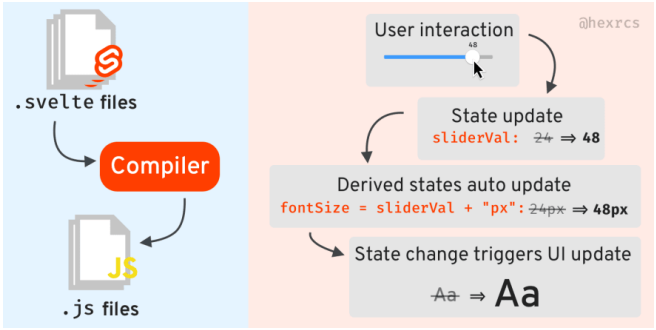
History

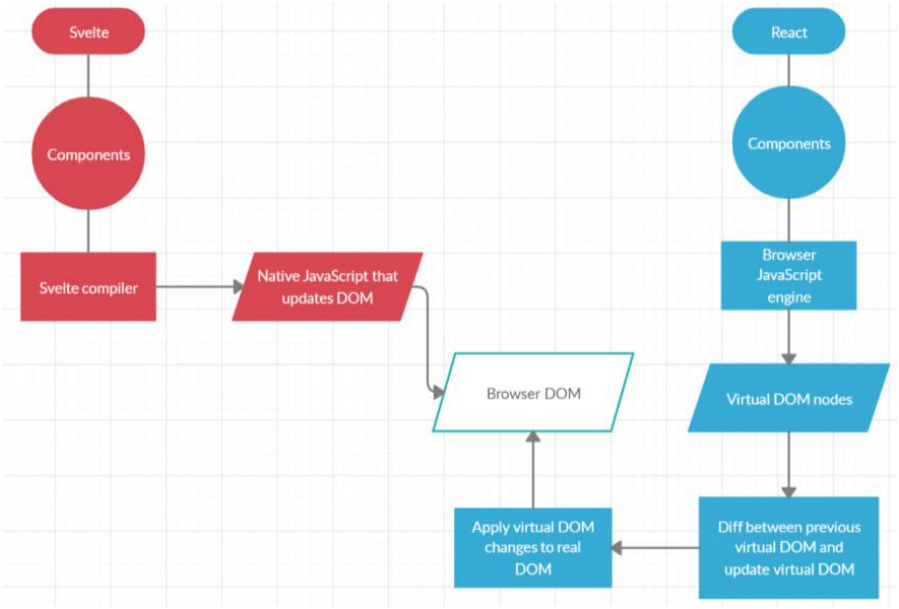
- Developer : **Rich Harris** (11/2016)
- Current version:
  - Svelte 3.18.2 (02/2020)
  - Sapper v0.27.9 (09/2019)
- Svelte -> Compiler and framework JS to create optimized, efficient and reactive apps (React, Vue & Angular)
- Sapper -> Framework JS (SPA routing, SSR, PWA, SEO…) based on Polka (express), Svelte & Rust (GatsbyJS, Next.js, Nuxt.js)
- Backends (+svelte in Frontend)
  - ASP.net (C#, .net core) -> https://github.com/NetCoreTemplates/svelte-spa
  - Laravel (PHP) -> https://dev.to/shuv1824/using-svelte-js-with-laravel-part-1-setting-up-laravel-application-with-svelte-36p4
  - Ruby on rails (Ruby) -> https://blog.usejournal.com/getting-started-with-svelte-and-rails-6-d8384c80ad6c
  - Django (Python) -> https://github.com/cdrappi/django-svelte/tree/master
  - Flask (Python) -> https://medium.com/@cabreraalex/svelte-js-flask-combining-svelte-with-a-simple-backend-server-d1bc46190ab9
- Mobile (native project): Svelte native (Svelte + NativeScript) https://svelte-native.technology/ (iOS & Android apps)
- Desktop apps (project) : Svelectron (Svelte + Electron) https://github.com/maxatwork/svelte-electron-template

Svelte-compiler

Compiler

- Rollup (JS bundler-loaders)
- Optimization & performance (wearable, IoT, smart TV/homes, control screens cars…)
- Use of modern JS (es6+, no babel)
- **No virtual DOM** required
  - Compiles your components (.svelte) into native JS modules during the building stage
  - Apps needs no dependencies to start
  - Component = f(state1, state2,…)
- Linter (good practices, accessibility…)
- CSS Preprocessor (Sass, postCSS) + TypeScript
- **SUPER** small bundle size (bundle.css & bundle.js ~ ko)
- **LESS** lines of code (40% less than React) &TRULY reactive

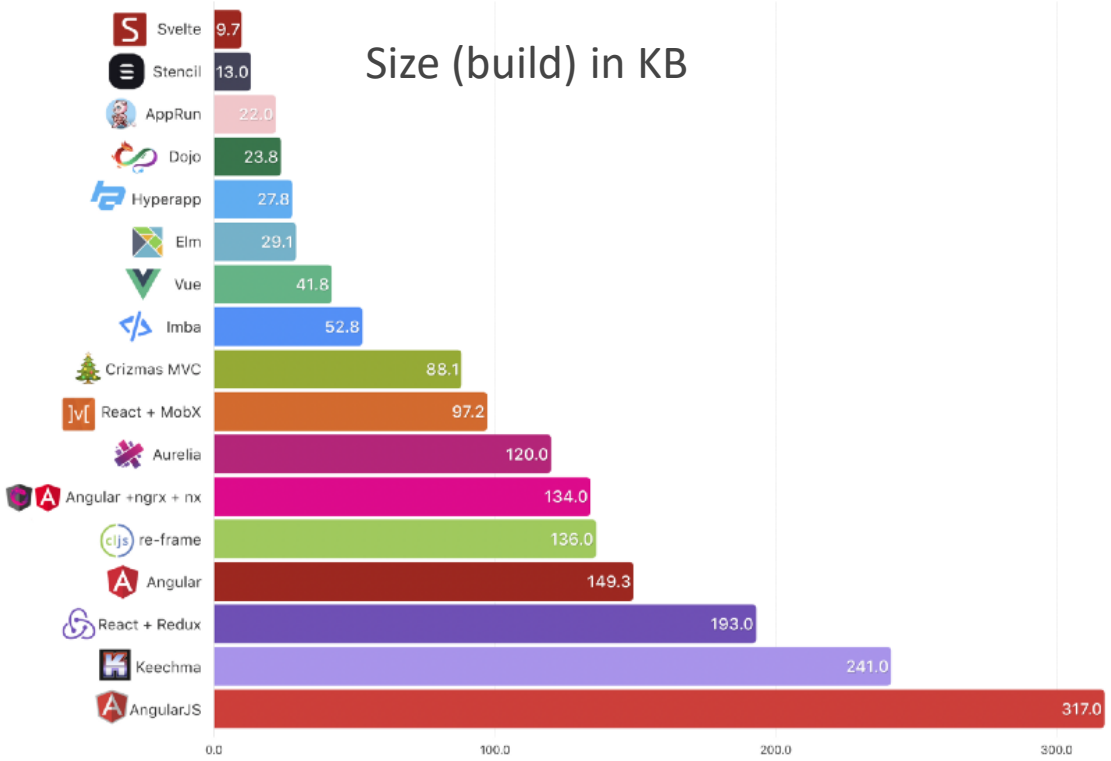


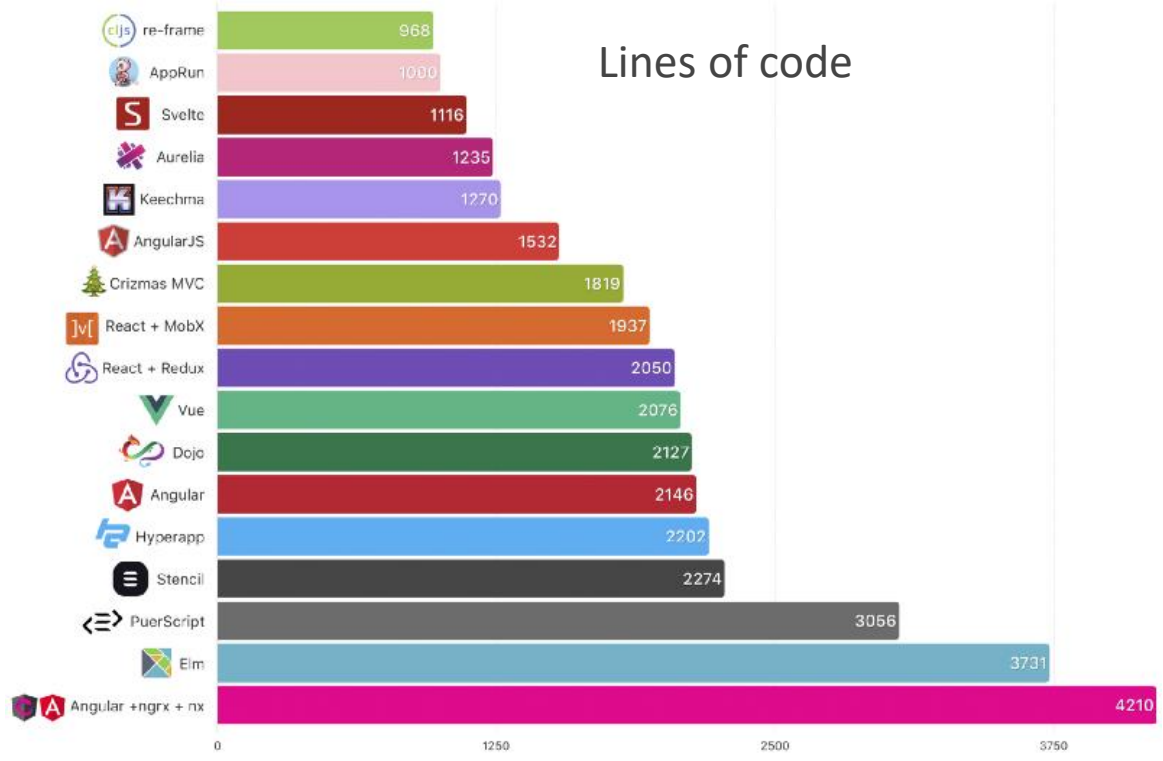


Svelte-tests

Tests

- 18 libraries/frameworks JS
- Conduit (light medium clone) https://demo.realworld.io/#/





Svelte-framework

Framework

- Young but gaining in popularity
- Dan Abramov (redux), Mike Bostock (d3.js)
- Component driven architecture (React) : props, state, HOC…
- Very light framework (3 Ko)
  - Hello word apps (1.16 Ko Svelte VS React 37.6 Ko)
- Code mainly in Vanilla JavaScript
- Very few features (svelte language)
- Anatomy of a Svelte Component

  <script /> (component logic)

  <script lang="typescript"> (`yarn add typescript --dev`)

  <style /> (scoped CSS)

  <style lang="scss"> (`yarn add svelte-preprocess node-sass --dev`)
- HTML (svelte template) -> JS in HTML + handlebars

```
<script>
  export let fontSize = "24px";
  let previewText = "";
</script>
```

Component Logic

```
<style>
  label {
    margin: 0.5rem 0;
  }
  textarea {
    width: 15rem;
    height: 15rem;
    font-family: "Courier New", Courier, monospace;
  }
</style>
```

Scoped CSS

```
<label for="preview">Preview:</label>
<textarea
  name="preview"
  id="preview"
  style="font-size: {fontSize}"
  bind:value={previewText} />
```

Svelte Template

@hexrcs

Svelte-setup

Local (dev) setup

- Template : npx degit sveltejs/template firstapp (https://github.com/sveltejs/template)
- Svelte extension for VS code
- Setup for SCSS (SASS) :

```
yarn add svelte-preprocess node-sass --dev
```

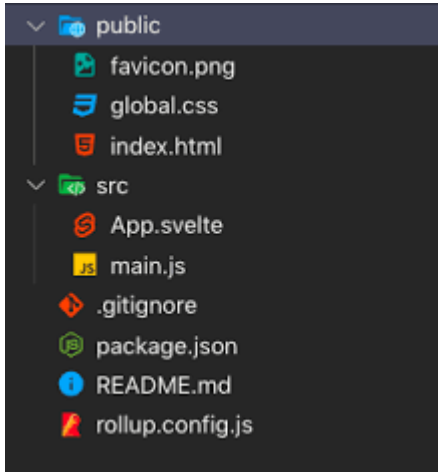
  - File **rollup.config.js**

```
import autoPreprocess from "svelte-preprocess";
plugins: [
 preprocess: autoPreprocess(),
    css: css => { css.write("public/build/bundle.css" }
]
```

  - Create file **svelte.config.js**

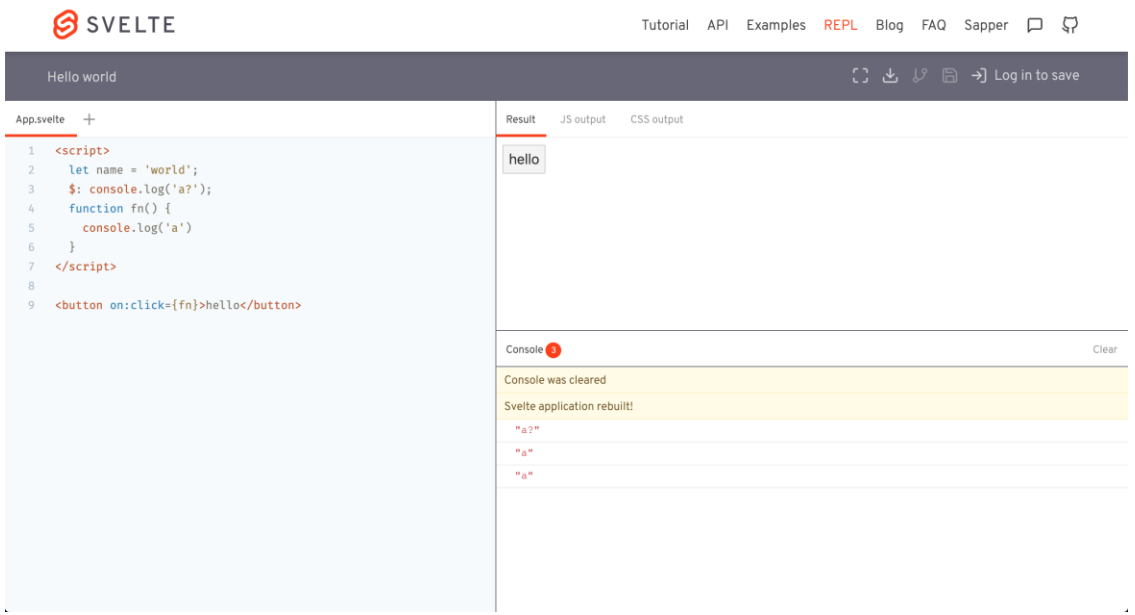
```
module.exports = { preprocess: autoPreprocess() };
```

- Execution:
  - run (dev) -> yarn run dev & run (build) -> yarn run build
  - App starts on localhost (port 5000 by default)



Online editor

- (Interpreter) REPL (Read–eval–print loop)
- https://svelte.dev/repl/hello-world?version=3.18.2
- Github info (sign up)



Svelte-features (1)

Features (1)

- **State**
  - no hooks (useState)
  - no setState (class component)
  - let declaration
  - reassignment (=)
- **Components**
  - Svelte files (.svelte)
  - Import create hierarchical relationship (parent -> child)
  - Passing data
    - Parent -> child (**props**) (export let propsName)
    - Parent -> descendants (**API context**)
    - setContext(key, val) & getContext(key)
- **HTML/CSS features**
  - **Boolean attribute** (flag=true/false)
    - <button disabled="{flag}">
  - **Conditionnal class** : <input class:alert="{flag}" />
  - Expression in template (like JSX) : {expression}

- **Template syntax**
  - **If-template** (conditionnal logic)

```
{#if expr1}
   HTML content 1
{:else if expr2}
   HTML content 2
{:else}
   HTML content 3
{/if}
```

  - **Loop-template**

Arrays

```
{#each fruits as fruit, index (index)}
   <li>{fruit}</li>
{/each}
```

Arrays of objects

```
{#each fruits as fruit, index (fruit.id)}
   <li>{fruit.name}: {fruit.price}</li>
{/each}
```

  - **Promises template**

Svelte-features (2)

Features (2)

- **Two-way binding**
  - **Value** (forms, media, sizes, files…)
    - bind:value={val}
    - bind:checked={val}
    - bind:group={val}
    - bind:currentTime={val}
    - bind:duration={val}
    - bind:paused={val}
    - bind:played={val}
    - bind:volume={val}
    - bind:offsetWidth={val} (read-only)
    - bind:offsetHeight={val}
    - bind:clientWidth={val}
    - bind:clientHeight={val}
    - bind:files={files}
  - **Props**
  - **DOM:** bind:this={val})
  - **Component**

- **Reactivity operator $** (like MobX & RxJS)
  - Listen for changes in the component state
  - Update other variables

```
$: double = count * 2
$: console.log('The count is', count)
$: if (expr) double = count * 2
$: {
     console.log('the count is', count)
     double = count * 2
     console.log('double the count is', double)
}
```

- **Events**
  - **DOM events** (on:click, on:change…)
  - **Events modifiers** (once, preventDefault…)
  - **Custom events**
  - **DOM event forwarding**
- **Slots**
  - **Slots & default slots**
  - **Named slots**
  - **Slots props**

Svelte-features (3)

Features (3)

- **Lifecycle**
  - **onMount**
  - **onDestroy**
  - **beforeUpdate**
  - **afterUpdate**
  - **tick**
    - Call it any time
    - Returns a promise that resolves ASAP any pending state changes have been applied to the DOM
    - (or immediately, if there are no pending state changes).
- **Motions**
  - **Tweened**

    Configure delay, duration, easing function…

    https://svelte.dev/tutorial/tweened
- **Animations**
  - **Transitions and animations CSS**

- **Promises-template**

```
<script>
  const fetchImage = (async () => {
    const response = await
      fetch('https://dog.ceo/api/breeds/image/random')
    if (response.ok) throw new Error(An error occurred!)
    return await response.json()
  })()
</script>

{#await fetchImage}
   <Loading />
{:then data}
   <img src={data.message} alt="Dog image" />
{:catch error}
   <p>An error occurred!</p>
{/await}
```

Svelte-features (4)

Features (4) – Store

- A store is any object that allows reactive access to a value via a simple store contract https://svelte.dev/docs#svelte_store
- Has **get()** for get the value of the store variable once
- Has **subscribe()** method, or prefix stores with **$** for auto-unsubscribes, prevents memory leaks!
- Types of stores
  - **Readable stores** (**readable()**)
    - Can't be updated from the outside (no set() & update() methods)
    - Fetch a resource from the network, API call, get data from the filesystem (using local Node.js server), Timer…
  - **Writable stores** (writable())
    - Has **set(newValue)**, **update(currentValue => f(currentValue))** methods
  - **Derived Stores**
    - A derived store allows you to create a new store value that depends on the value of an existing store.
  - **Custom stores**
    - Add custom functions based on set(), update(CRUD, reset()…)

Questions