Svelte

Rethinking Reactivity

Overview

Svelte is a new framework for front-end. You write your components using HTML, CSS and JavaScript (plus a few extra bits you can learn in under 5 minutes), and during your build process Svelte compiles them into tiny standalone JavaScript modules. By statically analysing the component template, we can make sure that the browser does as little work as possible.

Instead of using techniques like virtual DOM diffing, Svelte writes code that surgically updates the DOM when the state of your app changes.

Svelte is a radical new approach to building user interfaces. Whereas traditional frameworks like React and Vue do the bulk of their work in the browser, Svelte shifts that work into a compile step that happens when you build your app.

What problem do frameworks really solve?

The common view is that frameworks make it easier to manage the complexity of your code: the framework abstracts away all the fussy implementation details with techniques like virtual DOM diffing. But that's not really true. At best, frameworks move the complexity around, away from code that you had to write and into code you didn't.

Instead, the reason that ideas like React are so wildly and deservedly successful is that they make it easier to manage the complexity of your concepts. Frameworks are primarily a tool for structuring your thoughts, not your code.

Given that, what if the framework didn't actually run in the browser? What if, instead, it converted your application into pure vanilla JavaScript, just like Babel converts ES2016+ to ES5? You'd pay no upfront cost of shipping a hefty runtime, and your app would get seriously fast, because there'd be no layers of abstraction between your app and the browser.

Yes, I'm talking about Svelte

Reducing the amount of code you have to write is an explicit goal of Svelte. To illustrate, let's look at a very simple component implemented in React, Vue and Svelte.

In other words, it takes 442 characters in React, and 263 characters in Vue, to achieve something that takes 145 characters in Svelte. The React version is literally three times larger!

It's unusual for the difference to be quite so obvious — in my experience, a React component is typically around 40% larger than its Svelte equivalent. Let's look at the features of Svelte's design that enable you to express ideas more concisely:

Write Less Code

- All code is buggy. It stands to reason, therefore, that the more code you have to write the buggier your apps will be.

- Writing more code also takes more time, leaving less time for other things like optimisation, nice-to-have features, or being outdoors instead of hunched over a laptop.

- Yet while we obsess — rightly! — over performance numbers, bundle size and anything else we can measure, we rarely pay attention to the amount of code we're writing.

Readability is important

I'm certainly not claiming that we should use clever tricks to scrunch our code into the most compact form possible at the expense of readability. Nor am I claiming that reducing lines of code is necessarily a worthwhile goal.

Instead, I'm claiming that we should favour languages and patterns that allow us to naturally write less code.

Svelte is a compiler!

Being a compiler, Svelte can extend HTML, CSS, and JavaScript, generating optimal JavaScript code without any runtime overhead. To achieve this, Svelte extends vanilla web technologies in the following ways:

- It extends HTML by allowing JavaScript expressions in markup and providing directives to use conditions and loops, in a fashion similar to handlebars.
- It extends CSS by adding a scoping mechanism, allowing each component to define their own styles without the risk of clashing with other component's styles.
- It extends JavaScript by reinterpreting specific directives of the language to achieve true reactivity and ease component state management.

The compiler only intervenes in very specific situations and only in the context of Svelte components. Extensions to the JavaScript language are minimal and carefully picked in order to not break JavaScript syntax nor alienate developers. In fact, you will be mostly working with vanilla JavaScript.

Virtual DOM is pure overhead

In many frameworks, you build an app by creating render() functions.

An object represents how the page should now look. That object is the virtual DOM. Every time your app's state updates, you create a new one. The framework's job is to reconcile the new one against the old one, to figure out what changes are necessary and apply them to the real DOM.

You can't apply changes to the real DOM without first comparing the new virtual DOM with the previous snapshot.

Unlike traditional UI frameworks, Svelte is a compiler that knows at build time how things could change in your app, rather than waiting to do the work at run time.

Why do frameworks use the virtual DOM then?

It's important to understand that virtual DOM isn't a feature. It's a means to an end, the end being declarative, state-driven UI development.

Virtual DOM is valuable because it allows you to build apps without thinking about state transitions, with performance that is generally good enough. That means less buggy code, and more time spent on creative tasks instead of tedious ones.

But it turns out that we can achieve a similar programming model without using virtual DOM — and that's where Svelte comes in.

Rethinking reactivity

Functional

Avoid intricate stateful programs, using clean input/output functions over observable streams.

Less is more

ReactiveX's operators often reduce what was once an elaborate challenge into a few lines of code.

Async error handling

Traditional try/catch is powerless for errors in asynchronous computations, but ReactiveX is equipped with proper mechanisms for handling errors.

Concurrency made easy

Observables and Schedulers in ReactiveX allow the programmer to abstract away low-level threading, synchronization, and concurrency issues.

Where teams build faster, together

Create, share, and get feedback with collaborative sandboxes for rapid web development.

</> Start coding for free