Svelte

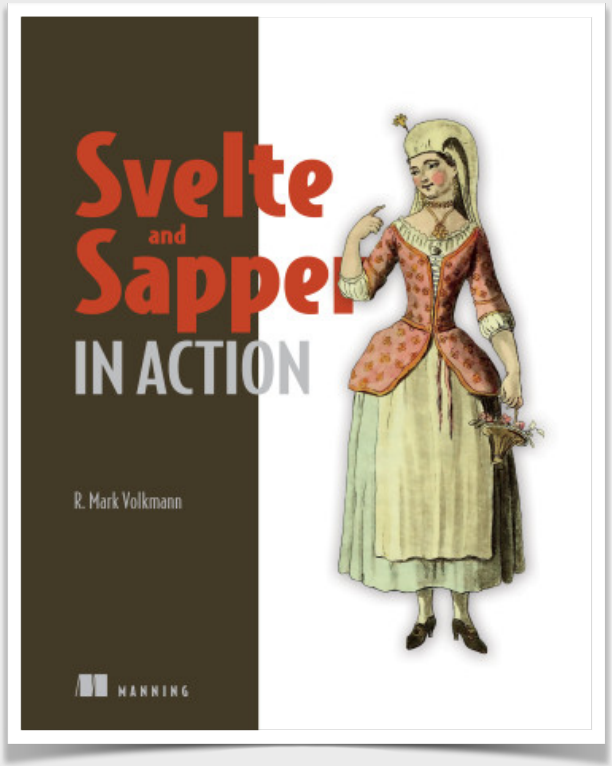
Web App Development Reimagined

R. Mark Volkmann
Object Computing, Inc.
https://objectcomputing.com
mark@objectcomputing.com
@mark_volkmann

Slides at https://github.com/mvolkmann/talks/ ... svelte

About Me

- Partner and Distinguished Software Engineer
  at Object Computing, Inc. in St. Louis, Missouri USA
- 42 years of professional software development experience
- Writer and teacher
- Blog at https://mvolkmann.github.io/blog/
- Author of Manning book “Svelte and Sapper in Action”
  - Sapper has been replaced by SvelteKit
    which has many of the same features
  - Much less expensive to buy directly through Manning
    when there is a sale rather than through Amazon
- Recently doing a lot of SwiftUI development



Svelte is ...

- An alternative to React, Vue, Angular, ...
- A compiler, not a runtime library
- Able to use JavaScript or TypeScript
- Developed by Rich Harris
  - also created of Ractive and Rollup
  - worked in graphics departments at “The Guardian” and “The New York Times”
  - now at Vercel developing Svelte full-time

https://svelte.dev

Being a Compiler

- Enables many things that cannot easily be done in other frameworks
  - eliminate use of virtual DOM
    - https://svelte.dev/blog/virtual-dom-is-pure-overhead
  - detect where component state and app state is used
    - so those parts of the DOM can be efficiently rebuilt when values change
  - support custom syntaxes
  - detect unused CSS
  - generate optimized JavaScript from `.svelte` files with no runtime library
    - only includes parts of Svelte framework that are used
- Can’t just copy ideas from Svelte into in other frameworks

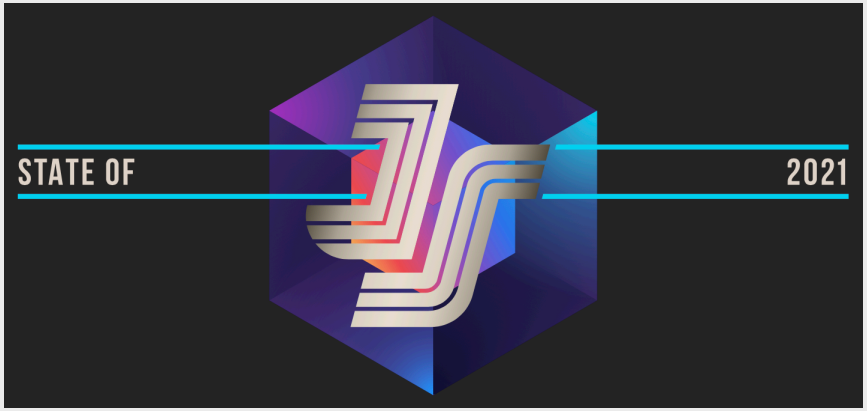
Benefits

- Less code to write
- File-based component definitions - JS, HTML, and CSS
- CSS scoped by default
- Easy component state management - **reactivity**
- Easy app state management - **stores**
- Reactive statements (`?:`) - like spreadsheet formulas
- Two-way data bindings - ex. between variable and **`<input>`**
- Built-in animations
- Small bundle sizes

2021 State of JS Survey

- https://2021.stateofjs.com/en-US/libraries/front-end-frameworks
- Results from 2019, 2020, and 2021

```
Angular satisfaction - 38% 42% 45%
  React satisfaction - 89% 88% 84%
 Svelte satisfaction - 88% 89% 90%
    Vue satisfaction - 87% 85% 80%

  Svelte interest - 67% 66% 68%
     Svelte Usage -  8% 15% 20%
Svelte awareness - 75% 87% 94%
```



`.svelte` File Contents

All parts are optional.

```
<script context="module">
</script>
<script>
  export let name;
</script>
<h1>Hello, {name}!</h1>
<style>
  h1 {
    color: red;
  }
</style>
```

think of as class-level; can export functions and constants; not often used

think of as instance-level; automatically exports component; cannot export anything else

declares a prop to be passed in

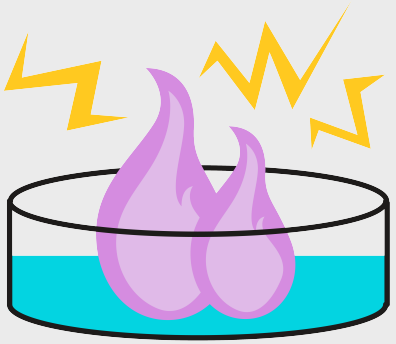
variables and functions are defined here

HTML can include Svelte logic and interpolations of expressions

automatically scoped to this component

Component State / Reactivity

- Just variables declared at top-level of **`script`** element
- Changes trigger updates to parts of DOM where used

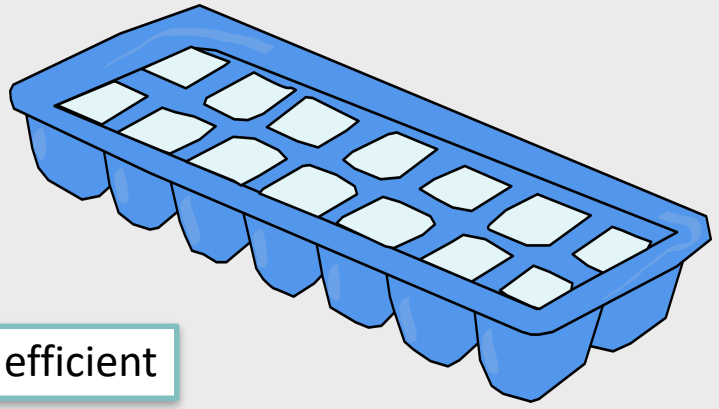


```
<script>
  let count = 0;
</script>

<div>
  <button disabled={count <= 0} on:click={() => count--}>-</button>
  <h1>{count}</h1>
  <button on:click={() => count++}>+</button>
</div>
```

- How much code would be required to implement this in other frameworks?

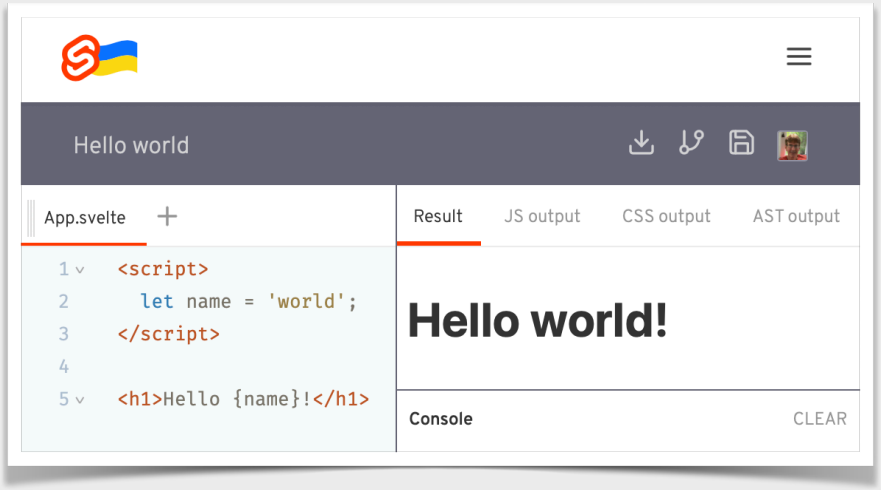
Reactivity of Arrays

- Changing variable “value” causes DOM updates
- But adding, removing, and changing array elements does not change array value
  - still refers to same array
- Three ways to handle
  - `myArr = myArr.concat(newValue);`
  - `myArr = [...myArr, newValue];`
  - `myArr.push(newValue); myArr = myArr;` most efficient



Svelte REPL

- Online **R**ead **E**val **P**rint **L**oop at https://svelte.dev/repl/
- Write and run Svelte apps without installing anything
- View generated JS and CSS
- Save to recall later
- Export to continue development outside REPL
- See my examples at https://mvolkmann.github.io/blog/ ... Svelte ... REPLS
- Demo Counter app in REPL

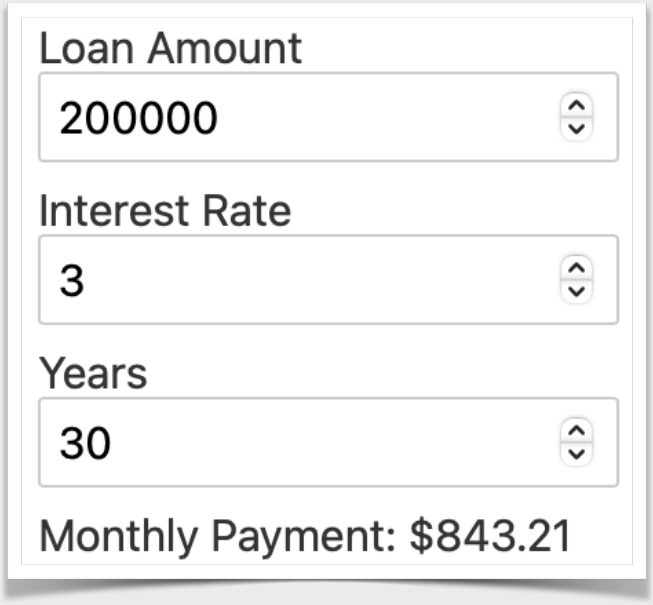


Reactive Statements

- Svelte interprets the JS label **$** to be a “reactive statement”
- Similar to spreadsheet formulas - ex. **=B3+C3**
- Code is re-executed every time a referenced variable changes
- Examples
  - `$: total = scores.reduce((acc, s) => acc + s, 0);`
  - `$: console.log('total =', total);`
  - `$: evaluateCart(cart, taxRate);`
  - `$: { ... }` block of code

Loan Calculator

- Calculates monthly payment from loan amount, interest rate, and number of years
- Great demo of using reactive statements
- Review code in REPL
- How much code would be required to implement this in other frameworks?



Conditional Logic in HTML

- Uses Mustache-like syntax
- `{#if some-condition}` opens with **#**

```
  HTML to render
{:else if other-condition}
  other HTML to render
{:else}
  more HTML to render
{/if}
```

continues with **:**

ends with **/**

Iteration in HTML

- `{#each arrExpr as elemName, index (keyExpr)}`
  `HTML to render for each element`
  `{:else}`
  `HTML to render if array is empty`
  `{/each}`
- `, index` is optional
- `(keyExpr)` is optional
  - include when array elements will be added, deleted, or reordered
- `{:else}` block is optional

Animations

- Supported by four packages
  - **`svelte/animate`** - **`flip`** function
  - **`svelte/easing`** - easing functions
  - **`svelte/motion`** - **`spring`** and **`tweened`** functions
  - **`svelte/transition`** - many directives and **`crossfade`** function
- All are CSS-based rather than JS-based
  - good performance because main thread is not blocked
- Can define custom transitions

animates changes to x/y position from old to new

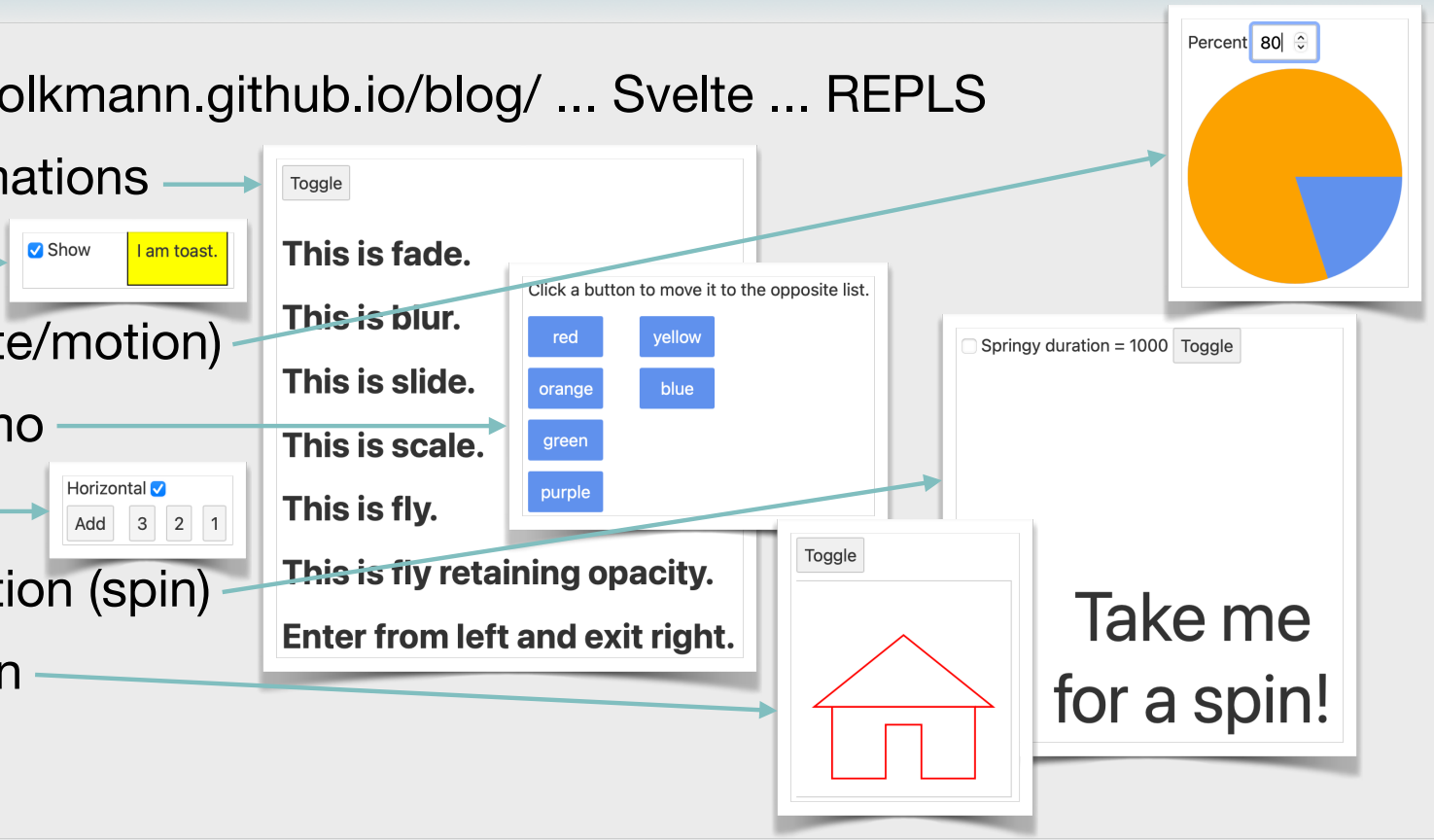
controls rate of change through an animation

return a **`writable`** store used to animate changes to a value

directives include **`blur`**, **`draw`**, **`fade`**, **`fly`**, **`scale`**, and **`slide`**

Animation Demos

- See https://mvolkmann.github.io/blog/ ... Svelte ... REPLS
- Transition Animations
- Toast
- Pie Chart (svelte/motion)
- Crossfade Demo
- Flip Animation
- Custom Transition (spin)
- Draw Animation



Events

- Events go from child component to parent
- Events have a name and optional data
- To dispatch from child

MyChild.svelte

```
<script>
  import {createEventDispatcher} from 'svelte';
  const dispatch = createEventDispatcher();
  dispatch('my-event', someData);
</script>
```

typically called when some user interaction occurs

- To listen in parent

MyParent.svelte

```
<script>
  import MyChild from './MyChild.svelte';
  function handleMyEvent(event) {
    const data = event.detail;
    ...
  }
</script>
<MyChild on:my-event={handleMyEvent}>
```

if no handler function is specified, the event propagates up

Demonstrated in TODO app ahead

Scoped CSS

- To achieve CSS scoping, Svelte ...
  - computes hash of all CSS in component
  - adds CSS class with name **`svelte-hash`** to all elements rendered by the component that are targeted by CSS
    - ex. **`<button>`** becomes **`<button class="svelte-4vhgyu">`**
  - adds same generated CSS class name to all CSS rules
    - ex. **`button { ... }`** becomes **`button.svelte-4vhgyu { ... }`**

You don't need to know this, but it is what happens behind the scenes.

Global CSS

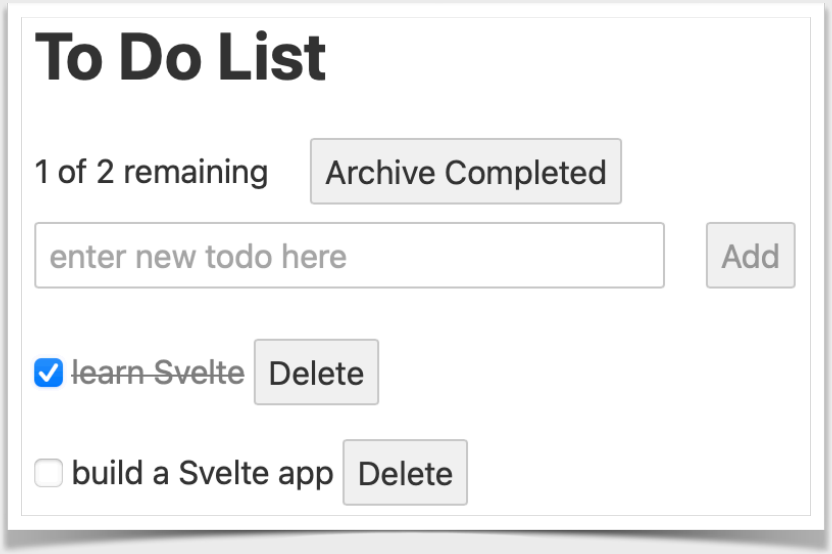
- Can define in a **`.css`** file and import into top component **`.svelte`** file
- Can avoid scoping in **`<style>`** of a component using **`:global(`*`selector`*`) { `*`properties`*` }`**
  - typically used to style nested components

```
<div class="my-component">
  <OtherComponent />
</div>

<style>
  .my-component :global(.other-component h1) {
    color: red;
  }
</style>
```

goal here is to change **`h1`** styling only for those rendered by **`OtherComponent`**

Todo App

- Classic app that demonstrates many Svelte features including
  - importing components
  - passing props
  - interpolation and **`#each`** in HTML
  - two-way data binding
  - reactivity
  - events
  - animation
  - scoped CSS
- Review code in REPL



App State / Stores

- Stores use publish/subscribe to share data between components
- Four kinds: **writable**, **readable**, **derived**, and **custom**
- Easiest approach is to define and export all in **`stores.js`**
- Can import in any components
- Subscribing and unsubscribing is automatic
  when store names are preceded by **`$`**

Writable Store Example

stores.js

```
import {writable} from 'svelte/store';
export const user = writable({
  firstName: '',
  lastName: ''
});
```

How much code would be required to implement this in other frameworks?

routes/index.svelte

```
<script>
  import {user} from '../stores.js';
  import Report from '../Report.svelte';
</script>
<input placeholder="First Name" bind:value={$user.firstName} />
<input placeholder="Last Name" bind:value={$user.lastName} />
<Report />
```

Report.svelte

```
<script>
  import {user} from './stores.js';
  function clear() {
    $user = {firstName: '', lastName: ''};
  }
</script>
<h1>Hello, {$user.firstName} {$user.lastName}.</h1>
<button on:click={clear}>Clear</button>
```

see “Writable Store” REPL

Readable Stores

- Cannot be changed from outside, but can change themselves
- Example
  - holds an array of objects describing dogs that come from a REST API
  - updated every five minutes

stores.js

```
import {readable} from 'svelte/store';

export const dogStore = readable([], async set => {
  const token = setInterval(() => {
    const res = await fetch('/dogs');
    const dogs = await res.json();
    set(dogs);
  }, 5 * 60 * 1000); // every 5 minutes

  return () => clearInterval(token);
});
```

Component Communication Options

Need	Solution
parent passes data to child	**props**
parent passes HTML and components to child	slots
child notifies parent, optionally including data	**events**
ancestor makes static data available to descendants	context
component shares data between all instances	module context
any component subscribes to and publishes data	**stores**

Actions & Slots

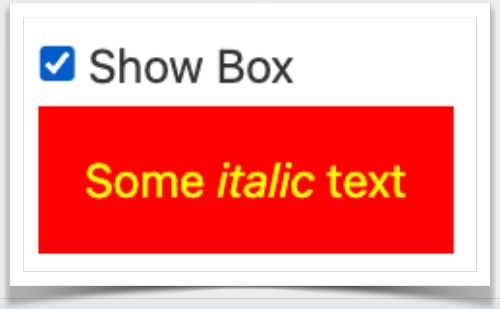
- **Actions** are functions that are called when an element is added to the DOM
  - often used to modify the DOM after an element appears
  - can also specify a code to run when the element is removed from the DOM
- **Slots** allow parent components to pass content to child components
- Both are demonstrated next

Actions & Slots Example

App.svelte

```
<script>
  import Box from './Box.svelte';
  let showBox = false
</script>

<div>
  <input type="checkbox" bind:checked={showBox} />
  Show Box
</div>

{#if showBox}
  <Box bgColor="red" fgColor="yellow">
    Some <i>italic</i> text
  </Box>
{/if}
```

slot content

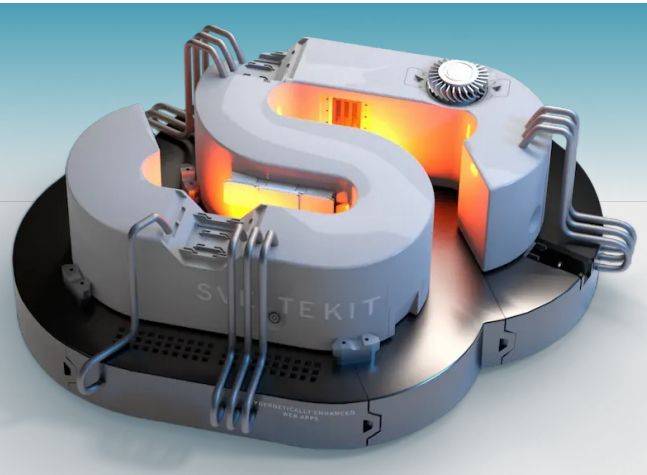


Box.svelte

```
<script>
  export let bgColor; // applied with an action
  export let fgColor = 'white'; // applied with style attribute

  function setBgColor(element) {
    console.log("added to DOM");
    element.style.backgroundColor = bgColor;
    return {
      destroy() {
        console.log("removed from DOM");
      }
    };
  }
</script>

<!-- Shows two approaches to applying style-related props. -->
<div class="box" style="color: {fgColor}" use:setBgColor>
  <slot /> <!-- content goes here -->
</div>

<style>
  .box {
    display: inline-block;
    padding: 1rem;
  }
</style>
```

see “Action and Slot Demo” REPL

SvelteKit



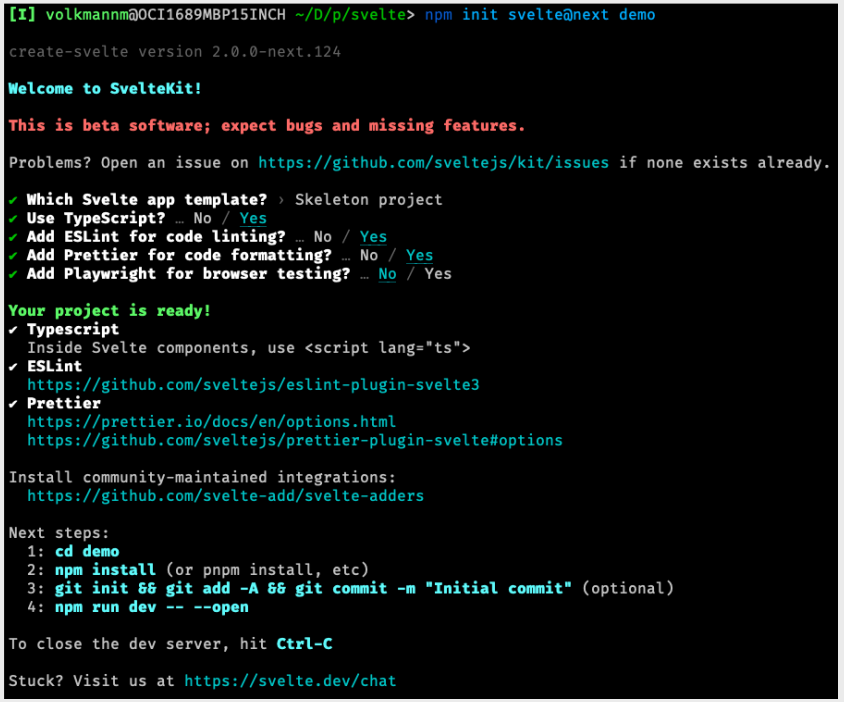
- Framework on top of Svelte that replaces Sapper
- Like Next for React and Nuxt for Vue
- Features
  - file-based page routing
  - file-based endpoints (APIs)
  - layouts - common header/footer/nav
  - error page
  - code splitting for JS and CSS
  - prefetching based on hover and focus
  - static pages and sites
  - hot module reloading using Vite
  - TypeScript, ESLint, and Prettier setup
  - adapters for deployment targets
    - official: static, Node, Netlify, Vercel
    - community: Begin, Cloudflare Workers, Deno, Firebase, and more

https://kit.svelte.dev

Creating a New Project

- **`npm init svelte@next` *`project-name`***
- Answer questions
  - Which Svelte app template?
    - SvelteKit demo app or Skeleton project (preferred)
  - Use TypeScript?
  - Add ESLint for code linting?
  - Add Prettier for code formatting?
  - Add Playwright for browser testing?

    alternative to Cypress



Running a Project

- Follow instructions that are output to install dependencies and run locally
  - `cd project-name`
  - `npm install`
  - `npm run dev -- --open`

SvelteKit Page Routing

- Pages and their URLs are described by directory and file names under **`src/routes`**
- File and directory names inside square brackets indicate that a path parameter will be captured

Page Routing Example

routes/index.svelte

```
<script>
  import '../global.css';
</script>
<nav>
  <a href="/page1">Page 1</a>
  <a href="/page2">Page 2</a>
</nav>
<main>
  <h1>Home Page</h1>
</main>
<style>
  a {
    color: white;
    font-size: 2rem;
    text-decoration: none;
  }
  main {
    padding: 1rem;
  }
  nav {
    display: flex;
    gap: 1rem;
    background-color: orange;
    padding: 1rem;
  }
</style>
```

global.css

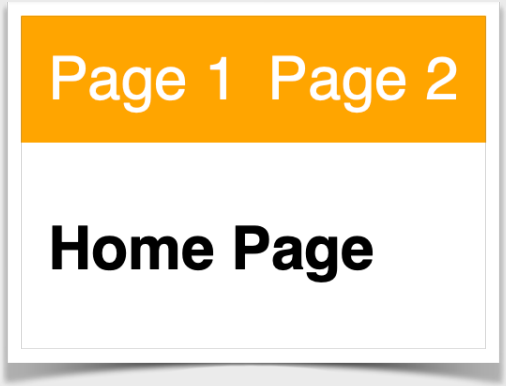
```
body {
  font-family: sans-serif;
  margin: 0;
}
```

routes/page1.svelte

```
<h1>First Page</h1>
```

routes/page2.svelte

```
<h1>Second Page</h1>
```



Wrap Up

- Developers like Svelte for many reasons,
  but the biggest reasons are that it
  reduces the amount of code that must be written
  and the code is easy to understand
- Delivering an excellent developer experience (DX)
  is not at the exclusion of good user experience (UX)
- Give Svelte a try and see if it
  simplifies web development for you!