

**Gather your party
with Svelte**

Who Are We?

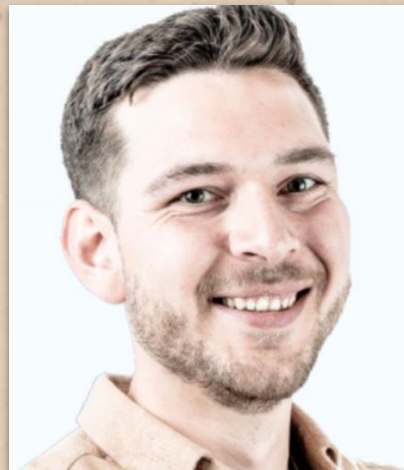


Dag Erik Løvgren

Frontend developer @ Miles Bergen

@Work: Loves clean, reusable code, and new technologies

@Home: Into RPGs, board games, and woodworking



Alexander Castillo

Fullstack developer @ Miles Bergen

@Work: Loves clean, reusable code, and new technologies & copy pasting

@Home: Into gaming, climbing, brewing beer and the great outdoors

Agenda



SVELTE

*CYBERNETICALLY ENHANCED
WEB APPS*

- Introduction to Svelte
- Workshop Description
- Workshop, let's a go!
- Conclusion

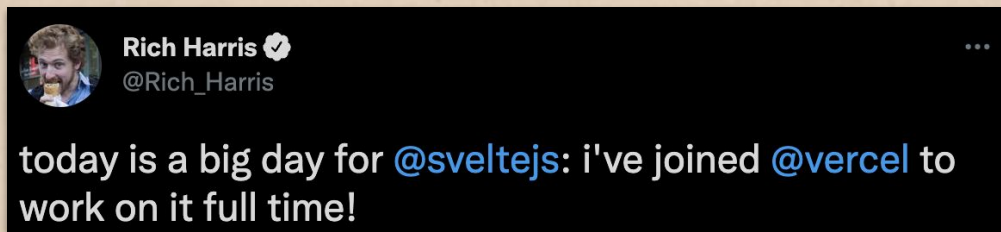
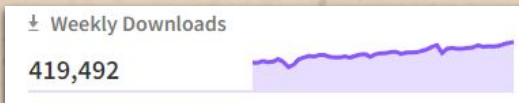
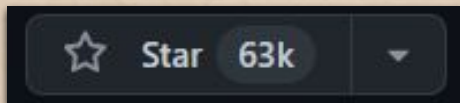
Yet Another Frontend Framework?!

- *Well, yes, it's an alternative*

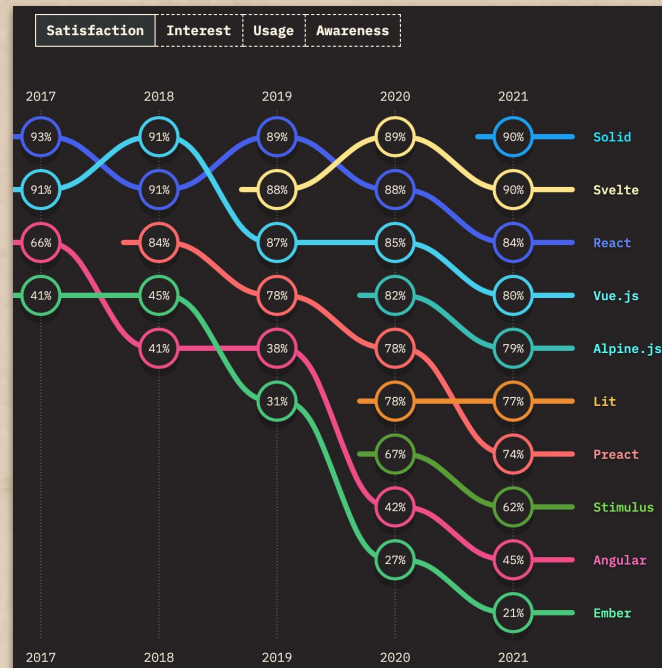
- Not really a framework, but a *compiler*
- Motivation for creating Svelte
 - Performance
 - Animation-heavy interfaces
 - Write less, do more
 - Embedded devices
- No Virtual DOM
 - But the Virtual DOM is fast, isn't it?
- Reactive
 - *"The best API is no API at all."*

Svelte in the Real World

- Vercel documented support
- Netlify documented support
- Sanity has a SvelteKit Template



State of JS 2021 Satisfaction:



Svelte Core Features

Covered in this workshop:

- Components
- Reactivity
- Blocks
- Bindings and Events
- Lifecycle
- Motion
- Transitions
- Animations

Not covered in this workshop:

- Stores
 - Subscribable global objects
 - <https://svelte.dev/tutorial/writable-stores>
- Actions
 - Element-level lifecycle functions
 - <https://svelte.dev/tutorial/actions>
- Context API
 - Inherit state in child components
 - <https://svelte.dev/tutorial/context-api>

Components

```
/// Usage:
<Button hint="< Click me" small={false}>Click me!</Button>
/// Button.svelte:
<script lang="ts">
  import ClickIcon from './ClickIcon.svelte';

  export let hint = '';
  export let small: boolean;
  export let disabled: boolean;
</script>

<button class="my-button" class:small {disabled} on:click>
  <slot />
  <ClickIcon />
</button>
<span>{hint}</span>

<style>
.my-button {
  border-radius: 4px;
  padding: 1rem 2rem;
}
.small {
  padding: 0.25rem;
}
</style>
```

CLICK ME! 

< CLICK ME

Reactivity

```
<script lang="ts">
  let count = 0;

  const increment = () => {
    count += 1;
  };

  $: sum = count * 2;
</script>

<button on:click={increment}>Click me to add 2</button>

<div>Clicked {count} {count === 1 ? 'time' : 'times'}</div>

<div>Sum: {sum}</div>

<style>
  * {
    margin-bottom: 1rem;
  }
</style>
```

CLICK ME TO ADD 2

CLICKED 0 TIMES

SUM: 0

Blocks

```

/// Usage:
<PortraitPreview id="portrait_11.jpg" tags={['Human', 'Bard']} />
/// PortraitPreview.svelte:
<script lang="ts">
  import { apiFetch } from '$lib/utils/api-fetch';

  export let id: string;
  export let tags: string[] = [];
</script>

<div class="container">
  {#if id}
    {#await apiFetch(`/api/portraits/${id}`)}
      Loading...
    {:then portrait}
      <img src={portrait} alt="Portrait" />
    {:catch error}
      Could not fetch portrait image.
    {/await}
  {/if}
</div>

{#each tags as tag (tag)}
  <span>({tag}) </span>
{/each}

```

LOADING...
(HUMAN) (BARD)

Bindings and Events

```

/// Usage:
<NameInput
  on:change={({ detail: { value } }) => console.log(value)}
  on:you-shall-not-enter={() => alert('No!!!')}
/>
/// NameInput.svelte:
<script lang="ts">
  import { createEventDispatcher } from 'svelte';
  const dispatch = createEventDispatcher();

  let name = 'world';

  const handleChange = () => {
    dispatch('change', { value: name });
  };
  const handleEnter = (event) => {
    if (event.key === 'Enter') {
      dispatch('you-shall-not-enter');
    }
  };
</script>

<input bind:value={name} on:input={handleChange}
on:keyup={handleEnter} />

<p>Hello {name}!</p>

```

WORLD

HELLO WORLD!

Lifecycle

```

<script lang="ts">
  import { onDestroy, onMount } from 'svelte';
  import { apiFetch } from '$lib/utils/api-fetch';

  let portrait: string;

  let counter = 0;
  const interval = setInterval(() => {
    counter += 1;
  }, 1000);

  onMount(async () => {
    portrait = await apiFetch('/api/portraits/portrait_15.jpg');
  });
  onDestroy(() => {
    clearInterval(interval);
  });
</script>

{#if portrait}
  <img src={portrait} alt="Portrait" />
{/if}

<p>This component has been mounted for {counter} seconds.</p>

```

THIS COMPONENT HAS BEEN MOUNTED FOR 0 SECONDS.

Motion

```

<script lang="ts">
  import { cubicOut } from 'svelte/easing';
  import { tweened } from 'svelte/motion';

  const MIN = 8;
  const MAX = 25;
  const DIFF = MAX - MIN;

  const strength = tweened(0, {
    duration: 800,
    easing: cubicOut,
  });
</script>

<button on:click={() => strength.set(MIN + Math.random() * DIFF)}>Randomize Strength</button>

<progress value={({strength - MIN} / DIFF) />
Strength: {Math.round($strength)}

```

RANDOMIZE STRENGTH

STRENGTH: 0

Transitions

```
<script lang="ts">
import { fly } from 'svelte/transition';

const scale = (node, { delay = 0, duration = 2000 }) => ({
  delay,
  duration,
  css: (t, u) => `
    font-size: calc(1em * ${u * 2 + 1});
    opacity: ${t};
  `;
});

let visible = false;
</script>

<button on:click={() => (visible = !visible)}>Say {visible ?
'goodbye' : 'hello'}</button>

{#if visible}
  <p in:fly={{ x: -200, duration: 2000 }} out:scale>Hello
there</p>
{/if}
```

SAY HELLO

Animations

```
<script lang="ts">
import { flip } from 'svelte/animate';
import { crossfade } from 'svelte/transition';

let todos = ['create a hero', 'level up', 'take first boss'];

// NB: This `crossfade` is a transition, `flip` is the animation
const [send, receive] = crossfade({
  duration: (d) => Math.sqrt(d * 200),
  fallback: () => ({
    duration: 600,
    css: (t) => `transform: scale(${t}); opacity: ${t}`;
  }),
});
</script>

{#each todos as todo (todo)}
  <button
    in:receive={{ key: todo }}
    out:send={{ key: todo }}
    animate:flip
    on:click={() => (todos = todos.filter((e) => e !== todo))}
  >
    {todo}
  </button>
{/each}
```

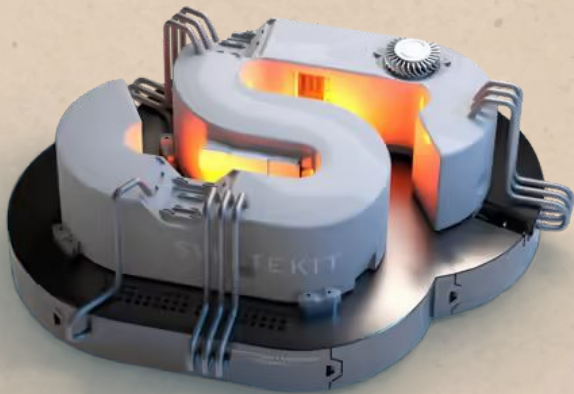
CREATE A HERO

LEVEL UP

TAKE FIRST BOSS

SvelteKit - There's another layer?

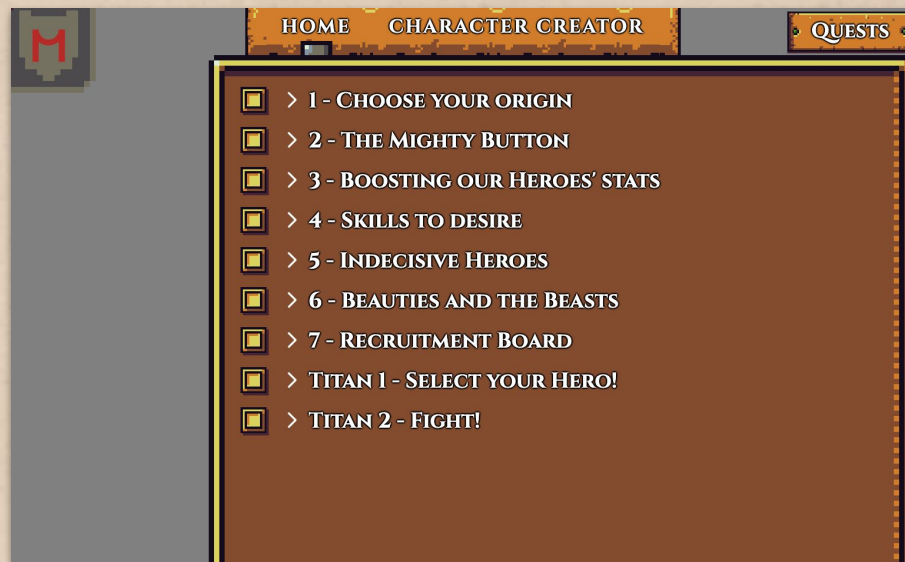
- Application framework for Svelte
- Similar to Next.js for React
- Still in beta
- Uses Vite
- Features:
 - Server-side Rendering (SSR)
 - Filesystem routing
 - SPA-like navigation
 - Code-splitting
 - Offline support
 - Hot Module Replacement (HMR)



Workshop Description

Glorious Quests! - Quest log

Alternative:
workshop-tasks.md



Glorious Quests! - Quests

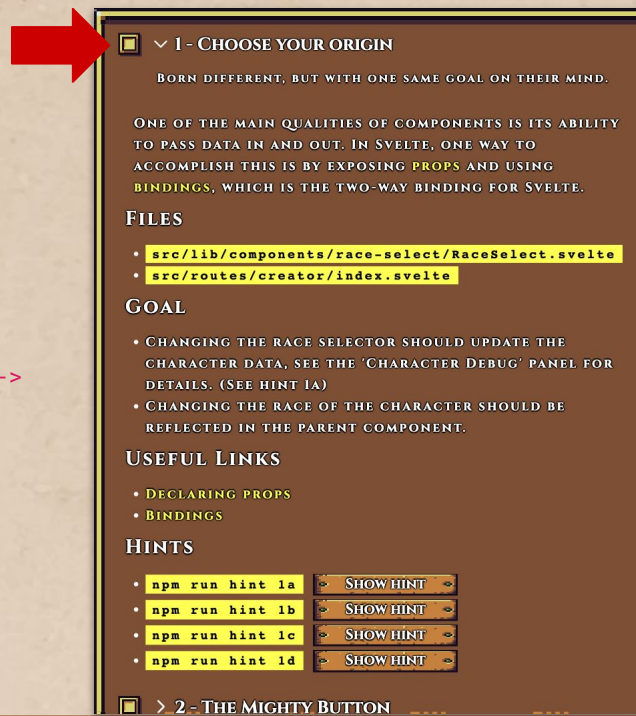
Quest files - unscoped

```
/**  
 * Quest 1 - Choose your origin  
 */
```

Quest areas - scoped

```
<!-- Quest 7 - Recruitment Board: Enter HTML markup here START -->  
<!-- Quest 7 - Recruitment Board: Enter HTML markup here END -->
```

Quest indicators



Glorious Quests! - Hints

```
npm run hint 1a
```

```
-----  
/ Hi, I will be your guide today! \  
\ You can call me Mr.Cowsy.      /  
-----
```

```
      ^__^  
      (oo)\_____  
      (--)\      )\\/  
           ||----w |  
           ||     ||
```

1 - CHOOSE YOUR ORIGIN

BORN DIFFERENT, BUT WITH ONE SAME GOAL ON THEIR MIND.

ONE OF THE MAIN QUALITIES OF COMPONENTS IS ITS ABILITY TO PASS DATA IN AND OUT. IN SVELTE, ONE WAY TO ACCOMPLISH THIS IS BY EXPOSING **PROPS** AND USING **BINDINGS**, WHICH IS THE TWO-WAY BINDING FOR SVELTE.

FILES

- `src/lib/components/race-select/RaceSelect.svelte`
- `src/routes/creator/index.svelte`

GOAL

- CHANGING THE RACE SELECTOR SHOULD UPDATE THE CHARACTER DATA, SEE THE 'CHARACTER DEBUG' PANEL FOR DETAILS. (SEE HINT 1A)
- CHANGING THE RACE OF THE CHARACTER SHOULD BE REFLECTED IN THE PARENT COMPONENT.

USEFUL LINKS

- [DECLARING PROPS](#)
- [BINDINGS](#)

HINTS

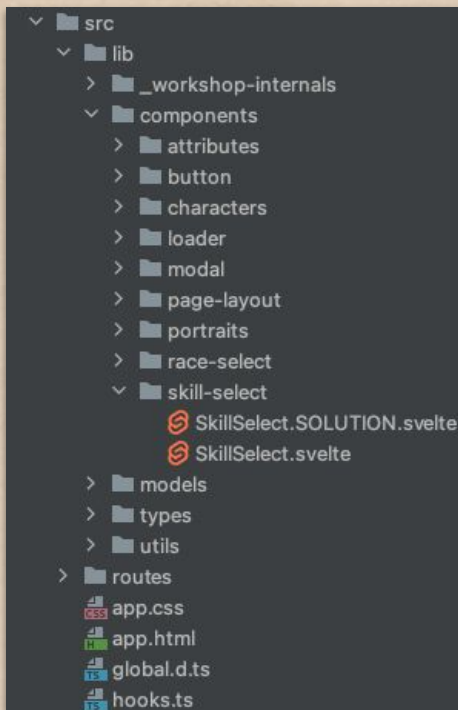
- `npm run hint 1a` [SHOW HINT](#)
- `npm run hint 1b` [SHOW HINT](#)
- `npm run hint 1c` [SHOW HINT](#)
- `npm run hint 1d` [SHOW HINT](#)

2 - THE MIGHTY BUTTON

Glorious Quests! - Solutions

Not THE solution, but A solution

Try first, but do take a peak if you feel stuck,
and have no more hints left.



```

  ▾ src
    ▾ lib
      > _workshop-internals
      ▾ components
        > attributes
        > button
        > characters
        > loader
        > modal
        > page-layout
        > portraits
        > race-select
        ▾ skill-select
          SkillSelect.SOLUTION.svelte
          SkillSelect.svelte
      > models
      > types
      > utils
    > routes
    app.css
    app.html
    global.d.ts
    hooks.ts

```

Let's get to Work!

Good luck, and feel free to ask questions!

Miles

tinyurl.com/miles-gypws

Start here  README.md

Conclusion

Questions?

Thanks for Us!



dag.erik.lovgren@miles.no



alexander.castillo@miles.no