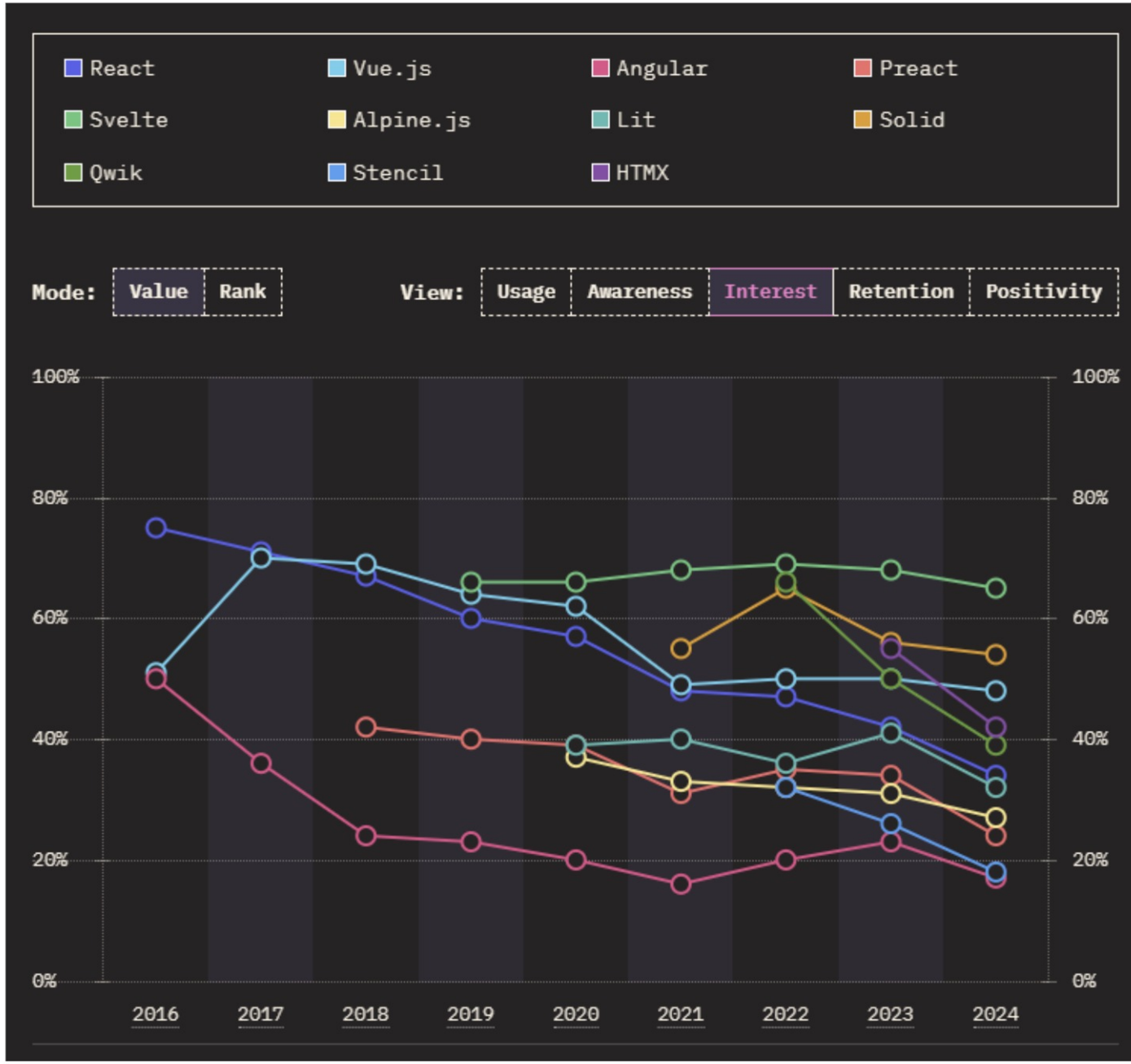
Intro to Sveltekit

A web framework

Overview:

0. Scaffolding an app with `sv`
1. Recommended Editor plugins
2. Core concepts
3. Loading data
4. Handing user data / forms
5. Composability
6. Lightning round
7. Adapters and deployment



State of JS 2024

Who am I?

Sam Peterson

🇺🇸 Minnesota, USA

Living in 🇩🇰 for 3 years

Full Stack since 2018

Passionate about:

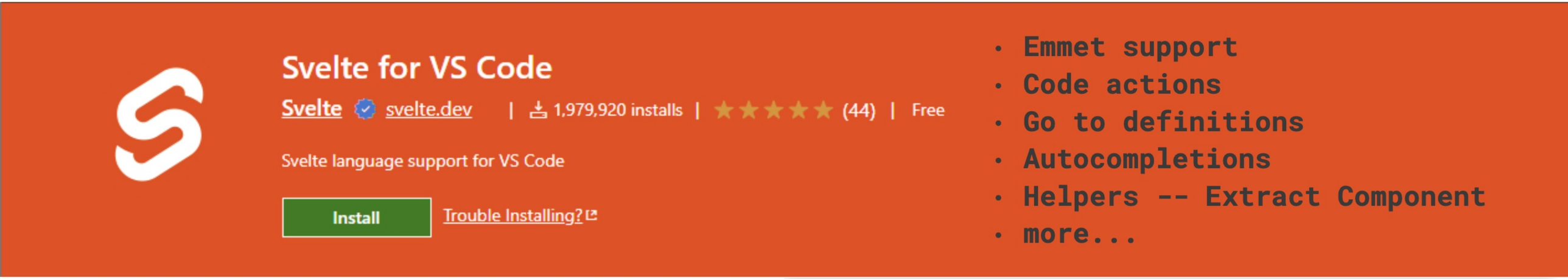
- traveling
- programming
- learning

Scaffolding an app

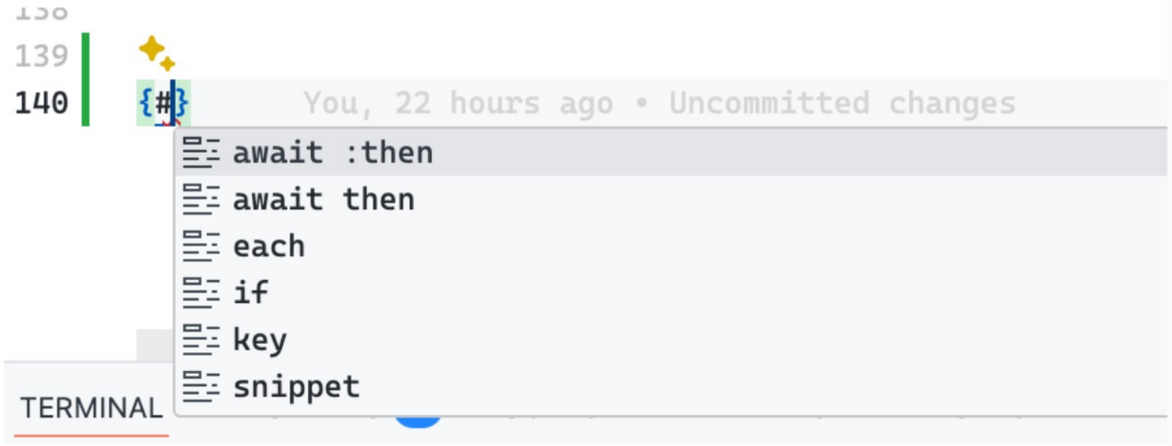
```
npx sv create
```

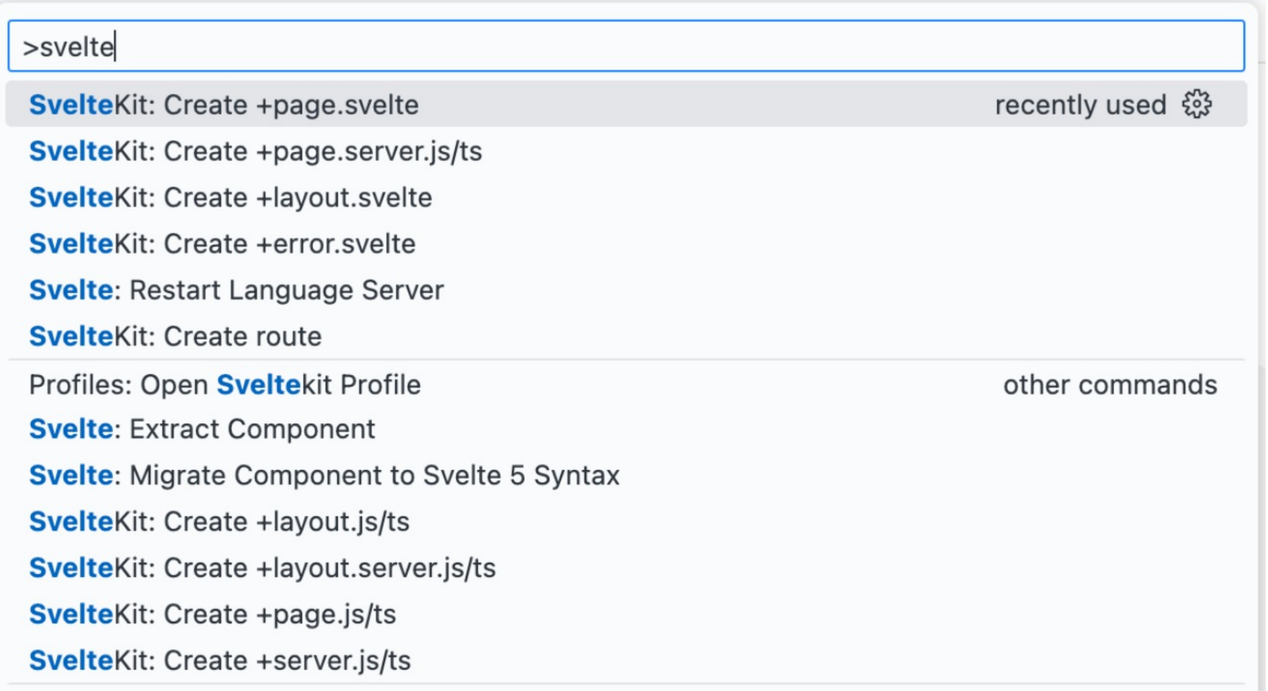
```
┌  Welcome to the Svelte CLI! (v0.8.0)
│
◇  Which template would you like?
│  SvelteKit minimal
│
◇  Add type checking with TypeScript?
│  Yes, using TypeScript syntax
│
◆  Project created
│
◇  What would you like to add to your project? (use arrow keys / space bar)
│  prettier, eslint, tailwindcss
│
◇  tailwindcss: Which plugins would you like to add?
│  typography, forms
│
◆  Successfully setup add-ons
│
◇  Which package manager do you want to install dependencies with?
│  deno
```

Editor plugin



- Emmet support
- Code actions
- Go to definitions
- Autocompletions
- Helpers -- Extract Component
- more...





What makes it different?

Svelte is a superset of HTML, not of Javascript

Svelte 5

Introduction of runes

- $state
- $derived
- $effect
- $props
- $bindable
- $inspect
- $host

Template syntax

- {#if...}...{/if}
- {#each ...}...{/each}
- {#await ...}
- {#snippet ...}
- {@render ...}
- {@html ...}
- bind:
- use:
- transition:

And more...

Loading patterns can be simple.

Remix

SVELTEKIT

Loading data

SVELTEKIT

src/routes/blog/[slug]/+page.ts

```ts
import type { PageLoad } from './$types';

export const load: PageLoad = ({ params }) => {
  return {
    post: {
      title: `Title for ${params.slug} goes here`,
      content: `Content for ${params.slug} goes here`
    }
  };
};
```

src/routes/blog/[slug]/+page.svelte

```svelte
<script lang="ts">
  import type { PageProps } from './$types';

  let { data }: PageProps = $props();
</script>

<h1>{data.post.title}</h1>
<div>{@html data.post.content}</div>
```

Loading data

app/routes/products.tsx

```tsx
import { json } from "@remix-run/node"; // or cloudflare/deno
import { useLoaderData } from "@remix-run/react";

export const loader = async () => {
  return json([
    { id: "1", name: "Pants" },
    { id: "2", name: "Jacket" },
  ]);
};

export default function Products() {
  const products = useLoaderData<typeof loader>();
  return (
    <div>
      <h1>Products</h1>
      {products.map((product) => (
        <div key={product.id}>{product.name}</div>
      ))}
    </div>
  );
}
```

app/routes/users.$userId.projects.$projectId.tsx

```tsx
import type { LoaderFunctionArgs } from "@remix-run/node";

export const loader = async ({
  params,
}: LoaderFunctionArgs) => {
  console.log(params.userId);
  console.log(params.projectId);
};
```

User data

The **<form>** HTML element represents a document section containing interactive controls for submitting information. Source MDN

src/routes/login/+page.svelte

```svelte
<form method="POST">
  <label>
    Email
    <input name="email" type="email">
  </label>
  <label>
    Password
    <input name="password" type="password">
  </label>
  <button>Log in</button>
</form>
```

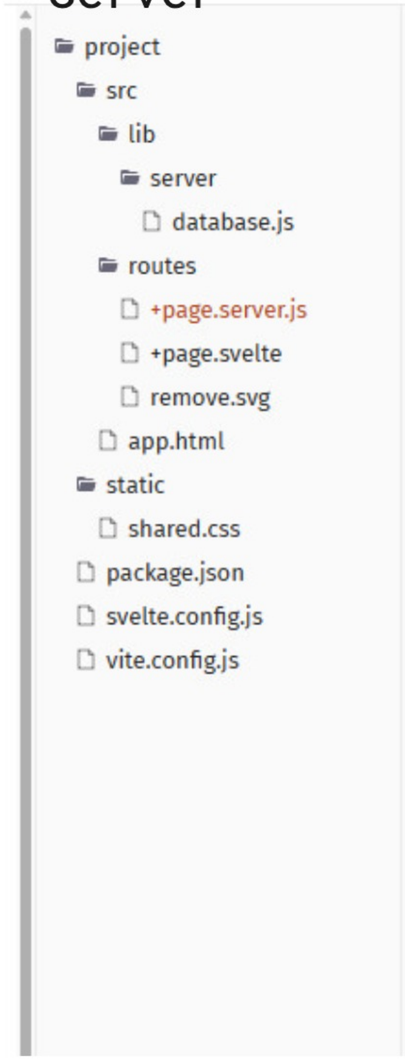
src/routes/login/+page.server.ts

```ts
import type { Actions } from './$types';

export const actions = {
  default: async (event) => {
    // TODO log the user in
  }
} satisfies Actions;
```

User data

Server




```
4     let id = cookies.get('userid');
5
6     if (!id) {
7       id = crypto.randomUUID();
8       cookies.set('userid', id, { path: '/' });
9     }
10
11    return {
12      todos: db.getTodos(id)
13    };
14  }
15
16  export const actions = {
17    create: async ({ cookies, request }) => {
18      const data = await request.formData();
19      db.createTodo(cookies.get('userid'), data.get('descript
20    },
21
22    delete: async ({ cookies, request }) => {
23      const data = await request.formData();
24      db.deleteTodo(cookies.get('userid'), data.get('id'));
25    }
26  };
27
```

Client

```
8   <form method="POST" action="?/create">
9     <label>
10      add a todo:
11      <input
12        name="description"
13        autocomplete="off"
14      />
15    </label>
16  </form>
17
18  <ul class="todos">
19    {#each data.todos as todo (todo.id)}
20      <li>
21        <form method="POST" action="?/delete">
22          <input type="hidden" name="id" value={todo.id} />
23          <span>{todo.description}</span>
24          <button aria-label="Mark as complete"></button>
25        </form>
26      </li>
27    {/each}
28  </ul>
```

Composability

Components:

App.svelte* Adjective.svelte* RUNES

```svelte
<script lang="ts">
  let props = $props<{adjective: string}>();
</script>

<p>this component is {props.adjective}</p>
```

App.svelte* Adjective.svelte* RUNES Result JS output CSS output AST output

```svelte
<script>
  import Adjective from './Adjective.svelte'
  let name = 'world';
</script>

<h1>Hello {name}!</h1>

<Adjective adjective="Cool" />
<Adjective adjective="Dry" />
<Adjective adjective="Simple" />
```

Hello world!

this component is Cool

this component is Dry

this component is Simple

Composability

Snippets:

App.svelte*

RUNES

Result | JS output | CSS output | AS

```svelte
<script>
  let { message = `it's great to see you!` } = $props();
</script>

{#snippet hello(name)}
  <p>hello {name}! {message}!</p>
{/snippet}

{@render hello('alice')}
{@render hello('bob')}
```

hello alice! it's great to see you!!

hello bob! it's great to see you!!

```svelte
{#snippet figure(image)}
  <figure>
    <img src={image.src} alt={image.caption} width={image.width}
    <figcaption>{image.caption}</figcaption>
  </figure>
{/snippet}

{#each images as image}
  {#if image.href}
    <a href={image.href}>
      {@render figure(image)}
    </a>
  {:else}
    {@render figure(image)}
  {/if}
{/each}
```

Lightning Round: CSS scoped by default

App.svelte*

```svelte
<script>
  import Button from "./Button.svel
</script>

<style>
  main {
    font-family: sans-serif;
    text-align: center;
  }
  button {
    border: black;
    background-color: rgba(
      200, 200, 235, 25%
    );
    border: solid black 1px;
    padding: 10px;
  }
</style>

<main>
  <h1>Hello Svelte</h1>
  <button>Different</button>
  <Button />
</main>
```

App.svelte* | Button.svelte* | RUNES

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

  function handleClick() {
    count += 1;
  }
</script>

<button on:click={handleClick}>
  Clicked {count} {count === 1 ? 'time' : 'times'}
</button>

<style>
  button {
    background: #ff3e00;
    color: white;
    border: none;
    padding: 8px 12px;
    border-radius: 2px;
  }
</style>
```

Lightning Round: Class helpers

Attribute:

App.svelte* RUNES Result

```svelte
<script>
  let items = ['One', 'Two', 'Three'];
</script>

<ul>
  {#each items as item}
    <li key={item}
      class={{
      'font-bold': true,
      'text-green-400': item.length <=3,
      'text-red-400': item.length > 3
      }}
      >
      {item}
    </li>
  {/each}
</ul>
```

```
One
Two
Three
```

Directive:

App.svelte* RUNES Result

```svelte
<script>
  let items = ['One', 'Two', 'Three'];
</script>

<ul>
  {#each items as item}
    <li key={item}
      class='font-bold'
      class:text-green-400={item.length <=3}
      class:text-red-400={item.length > 3}
      >
      {item}
    </li>
  {/each}
</ul>
```

```
One
Two
Three
```

Adapters

Allow SvelteKit apps to be deployed to a variety of locations.

svelte.config.js

```
import adapter from 'svelte-adapter-foo';

/** @type {import('@sveltejs/kit').Config} */
const config = {
  kit: {
    adapter: adapter({
      // adapter options go here
    })
  }
};

export default config;
```

Official adapters exist for a variety of platforms – these are documented on the following pages:

1. @sveltejs/adapter-cloudflare for Cloudflare Workers and Cloudflare Pages
2. @sveltejs/adapter-netlify for Netlify
3. @sveltejs/adapter-node for Node servers
4. @sveltejs/adapter-static for static site generation (SSG)
5. @sveltejs/adapter-vercel for Vercel

Additional community-provided adapters exist for other platforms.

Repository

Code demo and questions

Thanks for your attention

For making it to the end enjoy a funny photo of my dog.