SVELTE 3

WHAT IS SVELTE?

- Reactive frontend framework
- Fully component based
- Doesn't use a VDOM implementation
- Very fast with a tiny runtime
- Tries to reuses and repurposes existing syntax
- Heavily influenced by VueJS

FEATURES

- Uses single file components
- Components provide style encapsulation
- Support for PostCSS & SCSS
- Can be used with Webpack, Rollup or Parcel
- Partial support for Pug
- SSR via additional framework Sapper
- Support for Storybook

DISADVANTAGES

- Developer tools are quite barebones
- Typescript support poor
- No support for JSX or TSX
- No support for CSS-in-JS frameworks like SC
- No CLI like CRA or Vue CLI as of v3
- No bundler-free version as of v3
- A lot of outdated tutorials

IDEAL USE CASES

- Low powered devices
- Low bandwidth connections

ANATOMY OF A SVELTE COMPONENT

```
<script>
  js
</script>

<style>
  css
</style>

html
...
```

JAVASCRIPT

```
<script>
  // imports
  import ChildComponent from './ChildComponent.svelte';
  // props
  export let data;
  // function
  function getValue(id) {
    return data.find((entry) => entry.id === id);
  }
  // reactive var
  $: value = getValue('123');
  $: value2 = data.every((entry) => entry.id !== '123');
  // non reactive var
  const salutation = 'Howdy';
</script>
```

CSS

```
<style>
  .wrapper {
    margin: 0 auto;
    border: 1px solid var(--gray);
  }
</style>
```

```
<style>
  .wrapper.svelte-123456 {
    margin: 0 auto;
    border: 1px solid var(--gray);
  }
</style>
```

HTML

```
<div class="parent">
  <Component></Component>

  <Component/>

  <Component>
    <ChildComponent/>
  </Component>

  <h1 class="className">
    { title }
  </h1>
  <label for="field">Label</label>
</div>
```

```
<template lang="html">
  <div class="parent">
    .....
  </div>
</div>
```

SVELTE TEMPLATE LANGUAGE

ATTRIBUTES AND PROPS

```
<Component value={ value } text="text" text2=text />

<Component value={ x === 25 } value2="{ !isDefault }" />

<Component { hidden } />

<Component { ...allProps } />
```

CONTROL STRUCTURES

```
{#if id === 25}
  <p>Test</p>
{/if}

{#if id === 25}
  <p>Test</p>
{:else}
  <p>Test 2</p>
{/if}

{#if id === 25}
  <p>Test</p>
{:else if id === 52}
  <p>Test 2</p>
{:else}
  <p>Test 3</p>
```

LIST RENDERING

```
<ul>
  {#each list as listItem, i (listItem.id)}
    <li>{i} {listItem}</li>
  {/each}
</ul>

<ul>
  {#each list as listItem, i (listItem.id)}
    <li>{i} {listItem}</li>
  {:else}
    No entries
  {/each}
</ul>
```

SLOTS - REUSABILITY

```
<div class="v-card">
  <slot name="title">
    <h2>Default title</h2>
  </slot>
  <slot name="hobby">
    Musicals
  </slot>
</div>
```

```
<script>
  import VCard from './VCard.svelte';
</script>

<VCard>
  <div slot="title">
    <h2>Real title</h2>
  <div>
<VCard>
```

SLOTS - COMPOSITION

```
<div class="parent">
  <ChildComponent value={ value } />

  <ChildComponent value={ value } />
    <h2>{salutation}</h2>
  </ChildComponent>
</div>
```

```
<div class="child">
  <h1>Title {value}</h1>
  <slot>
    Optional default content
  </slot>
</div>
```

EVENTS

```
<a
  href="/"
  on:click={doSomething}
>
  Text
</a>

<a
  href="/"
  on:click|once|preventDefault|stopPropagation={doSomething}
>
  Text
</a>
```

EVENT HANDLING BETWEEN COMPONENTS

```
<div class="parent">
  <!-- Handle child event in parent -->
  <ChildComponent on:childeventname="funcInParent" />

  <!-- Relay child event to grandparent -->
  <ChildComponent on:childeventname />
</div>
```

```
<script>
  const dispatch = createEventDispatcher();

  function handleClick() {
    dispatch('childeventname');
  }
</script>
<div class="child">
  <a href="/" on:click|preventDefault={handleButtonClick}>Test</a>
</div>
```

COMMUNICATION BETWEEN COMPONENTS

- Props
- Events
- Two way data binding
- Context-API
- Store

DECLARATIVE PROMISE HANDLING

```
<script>
  const someAsyncOperation = new Promise((resolve, reject) => {
    setTimeout(() => resolve('12345'), 2500);
  });
</script>

<div class="component">
  {#await someAsyncOperation}
    <Spinner />
  {:then someAsyncOperationReturnValue}
    <MainComponent data={ someAsyncOperationReturnValue } />
  {:catch error}
    <span class="error">dang it - { error }</span>
  {/await}
</div>
```

OTHER FEATURES

- Life cycle methods (onMount, beforeUpdate, afterUpdate, onDestroy etc.)
- Built in transition- and animation-modules
- Store