SVELTE - QUICK TUTORIAL

SVELTE BRASIL

EXPRESSIONS

```
<p>2 + 2 = {2 + 2}</p>
<p>{isShowing ? 'NOW YOU SEE ME' : 'NOW YOU DON´T SEE ME'}</p>
<p>My e-mail is {email}</p>
<p>{user.name}</p>
<p>{cat + 's'}</p>
```

CONDITIONAL RENDER

```
{#if condition}
  <p>Condition is true</p>
{:else if otherCondition}
  <p>OtherCondition is true</p>
{:else}
  <p>Any Condition is true</p>
{/if}
```

AWAIT PROMISE IN TEMPLATE

```
{#await promise}
  <!-- promise is pending -->
  <p>waiting for the promise to resolve...</p>
{:then value}
  <!-- promise was fulfilled -->
  <p>The value is {value}</p>
{:catch error}
  <!-- promise was rejected -->
  <p>Something went wrong: {error.message}</p>
{/await}
```

RENDER HTML

```
{@html '<div>'}
content
{@html '</div>'}
{@html expression}
```

HANDLE EVENTS

```
<button on:click={handleClick}>Press me</button>

<!-- inline function -->
<button on:click={() => console.log('I was pressed')}>Press
me</button>

<!-- With modifiers -->
<button on:click|once={handleClick}>Press me</button>
<button on:submit|preventDefault={handleClick}>Press me</button>
```

Forwarding event

```
<script>
  import { createEventDispatcher } from "svelte";
  const dispatch = createEventDispatcher();
</script>
<!-- first param is the event name and the second is the value -->
<button on:click={() => dispatch('message', { text: 'Hello!' })}>
  Click to say hello
</button>
<!-- forwarding default event -->
<button on:click>Press me</button>
```

SIMPLE DATA BINDING

```
<MyLink href={href} title="My Site" color={myColor} />
```

For when props and variable match

```
<MyLink {href} title="My Site" color={myColor} />
```

Spreding props

```
<script>
  import MyLink from "./components/MyLink";
  let link = {
    href: "http://www.mysite.com",
    title: "My Site",
    color: "#ff3300"
  };
</script>
<MyLink {...link} />
```

TWO WAY DATA BINDING

```
<MyInput bind:value={value} />
```

For when props and variable match

```
<MyInput bind:value />
```

```
<select multiple bind:value={fillings}>
    <option value="Rice">Rice</option>
    <option value="Beans">Beans</option>
    <option value="Cheese">Cheese</option>
</select>
```

Binding groups

```
<!-- grouped radio inputs are mutually exclusive -->
<input type="radio" bind:group={tortilla} value="Plain" />
<input type="radio" bind:group={tortilla} value="Whole wheat" />

<!-- grouped checkbox inputs populate an array -->
<input type="checkbox" bind:group={fillings} value="Rice" />
<input type="checkbox" bind:group={fillings} value="Beans" />
```

Binding DOM element

```
<script> let myDiv </script>
<div bind:this={myDiv}></div>
```

RENDERING A LIST

```
<ul>
  {#each items as item}
    <li>{item.name} x {item.qty}</li>
  {:else}
    <li>Empty list</li>
  {/each}
</ul>
```

With index

```
{#each items as item, i}
  <li>{i + 1}: {item.name} x {item.qty}</li>
{/each}
```

With unique key

```
{#each items as item, i (item.id)}
  <li>{i + 1}: {item.name} x {item.qty}</li>
{/each}
```

By Demys Cota

SVELTE - QUICK TUTORIAL

SVELTE BRASIL

USING SLOT

```
<!-- Widget.svelte -->
<div>
  <slot>Default content</slot>
</div>

<!-- App.svelte -->
<Widget />
<Widget>
  <p>I  changed the default content</p>
</Widget>
```

Multiple slot

```
<!-- Widget.svelte -->
<div>
  <slot name="header">No header was provided</slot>
  <slot>
    <p>Some content between header and footer</p>
  </slot>
  <slot name="footer" />
</div>

<!-- App.svelte -->
<Widget>
  <h1 slot="header">Hello</h1>
  <p slot="footer">Copyright (c) 2020 Svelte Brazil</p>
</Widget>
```

Expose values

```
<!-- FancyList.svelte -->
<ul>
  {#each items as item}
    <li class="fancy">
      <slot name="item" {item} />
    </li>
  {/each}
</ul>

<!-- App.svelte -->
<FancyList {items}>
  <div slot="item" let:item>{item.text}</div>
</FancyList>
```

ANIMATIONS

```
<script>
  import { flip } from "svelte/animate";
  import { quintOut } from "svelte/easing";
  let list = [1, 2, 3];
</script>

{#each list as n (n)}
  <div animate:flip={{ delay: 250, duration: 250, easing: quintOut
}}>{n}</div>
{/each}
```

REACTIVE EXPRESSIONS

```
<script>
 export let num
 // we don't need to declare `squared` and `cubed`
 // — Svelte does it for us
 $: squared = num * num
 $: cubed = squared * num
</script>
```

CLASS BINDING

```
<!-- These are equivalent -->
<div class={active ? 'active' : ''}>...</div>
<div class:active>...</div>

<!-- Shorthand, for when name and value match -->
<div class:active>...</div>

<!-- Multiple class toggles can be included -->
<div class:active class:inactive={!active} class:isAdmin>
...
</div>
<div class="active {variableClass}">...</div>
```

LIFECYCLE

beforeUpdate	**onMount**
afterUpdate	**onDetroy**

STORE

```
<!--store.js -->
import { writeable } from 'svelte/store'
export const myNumber = whiteable(0)

<!--app.svelte-->
<script>
    import {myNumber} from './store.js'
</script>

<input type="number" bind:value={$myNumber} />
<label>Current value is {$myNumber}</label>
```

Others stores

Readable **Derived** **Custom**

TRANSITIONS

```
<script>
  import { fade } from "svelte/transition";
</script>

{#if condition}
  <div transition:fade={{ delay: 250, duration: 300 }}>fades in
and out</div>
{/if}
```

Others transitions

Blur **Fly** **Slide** **Scale** **Draw**

REACTIVE STATEMENT

```
<script>
 let count = 0
 $: if (count >= 10) {
   alert(`count is dangerously high!`)
   count = 9
 }
</script>
```

By Demys Cota