SVELTE • TEMPLATE SYNTAX

{#await ...}

```
{#await expression}...{:then name}...{:catch name}...{/await}
```

```
{#await expression}...{:then name}...{/await}
```

```
{#await expression then name}...{/await}
```

```
{#await expression catch name}...{/await}
```

Await blocks allow you to branch on the three possible states of a `Promise` — pending, fulfilled or rejected.

```
{#await promise}
  <!-- promise is pending -->
  <p>waiting for the promise to resolve...</p>
{:then value}
  <!-- promise was fulfilled or not a Promise -->
  <p>The value is {value}</p>
{:catch error}
  <!-- promise was rejected -->
  <p>Something went wrong: {error.message}</p>
{/await}
```

> During server-side rendering, only the pending branch will be rendered.
>
> If the provided expression is not a `Promise`, only the `:then` branch will be rendered, including during server-side rendering.

The `catch` block can be omitted if you don't need to render anything when the promise

```
  <!-- promise is pending -->
  <p>waiting for the promise to resolve...</p>
{:then value}
  <!-- promise was fulfilled -->
  <p>The value is {value}</p>
{/await}
```

If you don't care about the pending state, you can also omit the initial block.

```
{#await promise then value}
  <p>The value is {value}</p>
{/await}
```

Similarly, if you only want to show the error state, you can omit the `then` block.

```
{#await promise catch error}
  <p>The error is {error}</p>
{/await}
```

> You can use `#await` with `import(...)` to render components lazily:
>
> ```
> {#await import('./Component.svelte') then { default: Component }}
>   <Component />
> {/await}
> ```

Edit this page on GitHub