SVELTE • LEGACY APIS

<slot>

ON THIS PAGE

In Svelte 5, content can be passed to components in the form of snippets and rendered using render tags.

In legacy mode, content inside component tags is considered *slotted content*, which can be rendered by the component using a `<slot>` element:

App.svelte

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

<Modal>This is some slotted content</Modal>
```

Modal.svelte

```
<div class="modal">
  <slot></slot>
</div>
```

> If you want to render a regular `<slot>` element, you can use `<svelte:element this={'slot'} />` .

Named slots

A component can have *named* slots in addition to the default slot. On the parent side, add a `slot="..."` attribute to an element, component or `<svelte:fragment>` directly inside the

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

  let open = true;
</script>

{#if open}
  <Modal>
    This is some slotted content

    <div slot="buttons">
      <button on:click={() => open = false}>
        close
      </button>
    </div>
  </Modal>
{/if}
```

On the child side, add a corresponding `<slot name="...">` element:

Modal.svelte

```
<div class="modal">
  <slot></slot>
  <hr>
  <slot name="buttons"></slot>
</div>
```

Fallback content

If no slotted content is provided, a component can define fallback content by putting it inside the `<slot>` element:

```
<slot>
  This will be rendered if no slotted content is provided
</slot>
```

Slots can be rendered zero or more times and can pass values *back* to the parent using props. The parent exposes the values to the slot template using the `let:` directive.

The usual shorthand rules apply — `let:item` is equivalent to `let:item={item}`, and `<slot {item}>` is equivalent to `<slot item={item}>`.

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

<!-- App.svelte -->
<FancyList {items} let:prop={thing}>
  <div>{thing.text}</div>
</FancyList>
```

Named slots can also expose values. The `let:` directive goes on the element with the `slot` attribute.

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

<slot name="footer" />

<!-- App.svelte -->
<FancyList {items}>
  <div slot="item" let:item>{item.text}</div>
  <p slot="footer">Copyright (c) 2019 Svelte Industries</p>
```

PREVIOUS
on:

NEXT
$$slots