Slots in Vue3

Slots allow you to pass content (HTML/components) from a parent to a child component, while keeping the layout defined in the child.

OR

Slots allow a component to accept template content from its parent and render it in a specific place.

Dear child component, here's the content I want you to show in the empty space you prepared.

Types of Slots in Vue 3

1. Default Slot

The simplest form of a slot that passes unnamed content from parent to child.

Use when you need to insert basic content into a component without needing to differentiate sections.

Example:

```
<!-- BaseLayout.vue -->
<template>
  <div class="container">
    <slot></slot> <!-- Default slot -->
  </div>
</template>

<!-- App.vue -->
<BaseLayout>
  <p>This content will be rendered inside the slot.</p>
</BaseLayout>
```

2. Named Slot

Named slots let you define multiple slots in a component, each with a specific name.

Use when your component has multiple content sections (e.g., header, footer, sidebar).

Example:

```
<!-- Card.vue -->
<template>
  <div class="card">
    <header><slot name="header" /></header>
    <main><slot /></main> <!-- default slot -->
    <footer><slot name="footer" /></footer>
  </div>
</template>
```

```
<!-- App.vue -->
<Card>
  <template #header><h1>Title</h1></template>
  <p>Main content goes here.</p>
  <template #footer><small>Footer text</small></template>
</Card>
```

3. Scoped Slot

Scoped slots allow child components to expose data to the parent via the slot's scope.

Use when the parent needs to render slot content using data from the child component.

Example:

```
<!-- ItemList.vue -->
<template>
  <ul>
    <li v-for="item in items" :key="item.id">
      <slot :item="item" />
    </li>
  </ul>
</template>

<script setup>
const items = [
  { id: 1, name: 'Apple' },
  { id: 2, name: 'Banana' }
]
</script>

<!-- App.vue -->
<ItemList v-slot="{ item }">
  <strong>{{ item.name }}</strong>
</ItemList>
```

4. Conditional Slot

A slot that is rendered only when passed or when a condition is met.

Use to conditionally render optional sections (e.g., sidebar, tooltip).

Example:

```
<!-- Panel.vue -->
<template>
  <div>
    <slot />
    <div v-if="$slots.sidebar">
```

```
      <slot name="sidebar" />
    </div>
  </div>
</template>

<!-- App.vue -->
<Panel>
  <p>Main panel content.</p>
  <template #sidebar><p>This is a sidebar</p></template>
</Panel>
```

5. Dynamic Slot Name

Allows slot names to be dynamically defined using computed or reactive values.

Useful when you need to render different slot content dynamically.

Example:

```
<!-- DynamicSlot.vue -->
<template>
  <slot :name="slotName" />
</template>

<script setup>
import { ref } from 'vue'
const slotName = ref('one')
</script>

<!-- App.vue -->
<DynamicSlot>
  <template #one><p>Slot One</p></template>
  <template #two><p>Slot Two</p></template>
</DynamicSlot>
```

6. Renderless/Functional Component Using Scoped Slot

A component that provides data and behavior but no UI, relying on scoped slots to render.

Use for logic-only components (e.g., toggle, fetch, pagination logic).

Example:

```
<!-- Toggle.vue -->
<script setup>
import { ref } from 'vue'

const isOn = ref(false)
const toggle = () => (isOn.value = !isOn.value)
```

```
defineExpose({ isOn, toggle })
</script>

<template>
  <slot :isOn="isOn" :toggle="toggle" />
</template>

<!-- App.vue -->
<Toggle v-slot="{ isOn, toggle }">
  <button @click="toggle">
    {{ isOn ? 'ON' : 'OFF' }}
  </button>
</Toggle>
```