Built-in Components in vue3

Vue 3 provides some special components that help you control rendering, transitions, async loading, and more. They are part of the core Vue runtime and can be used like any other component in templates.

List of Vue 3 Built-in Components

Component	Purpose
`<component>`	Dynamically renders a component based on a variable
`<transition>`	Adds enter/leave animations when a component or element appears/disappears
`<transition-group>`	Adds transition to a list of elements/components
`<keep-alive>`	Caches inactive component instances
`<slot>`	Placeholder for child content
`<teleport>`	Renders content in a different part of the DOM
`<suspense>`	Waits for async components or templates to load

1. `<component :is="...">`

Switch between components dynamically. Think of this like a TV remote we can switch channels (components) anytime.

```
<!-- App.vue -->
<script setup>
import { ref } from 'vue'
import HelloWorld from './HelloWorld.vue'
import GoodbyeWorld from './GoodbyeWorld.vue'

const current = ref('HelloWorld')
const components = { HelloWorld, GoodbyeWorld }
</script>

<template>
  <button @click="current = current === 'HelloWorld' ? 'GoodbyeWorld' :
'HelloWorld'">
    Switch Component
  </button>

  <component :is="components[current]" />
</template>
```

2. `<transition>`

Add animation when showing/hiding. Like a slide-in or fade effect when a message appears.

```
<script setup>
import { ref } from 'vue'
const show = ref(false)
</script>

<template>
  <button @click="show = !show">Toggle</button>

  <transition name="fade">
    <p v-if="show">Hello, I fade in and out!</p>
  </transition>
</template>

<style>
.fade-enter-active, .fade-leave-active {
  transition: opacity 0.5s;
}
.fade-enter-from, .fade-leave-to {
  opacity: 0;
}
</style>
```

3. `<transition-group>`

Animate list items. Like animating a list of tasks, when you add/remove them.

```
<script setup>
import { ref } from 'vue'
const items = ref(['Vue', 'React', 'Angular'])

function addItem() {
  items.value.push('Svelte')
}
</script>

<template>
  <button @click="addItem">Add Item</button>

  <transition-group name="list" tag="ul">
    <li v-for="(item, index) in items" :key="index">{{ item }}</li>
  </transition-group>
</template>

<style>
.list-enter-active, .list-leave-active {
  transition: all 0.5s;
}
.list-enter-from {
  opacity: 0;
  transform: translateY(20px);
}
</style>
```

4. `<keep-alive>`

Save memory & keep form values. Imagine you filled a form, moved to another tab, and came back form is still there.

```
<!-- App.vue -->
<script setup>
import { ref } from 'vue'
import TabA from './TabA.vue'
import TabB from './TabB.vue'

const currentTab = ref('TabA')
const tabs = { TabA, TabB }
</script>

<template>
  <button @click="currentTab = 'TabA'">Tab A</button>
  <button @click="currentTab = 'TabB'">Tab B</button>

  <keep-alive>
    <component :is="tabs[currentTab]" />
  </keep-alive>
</template>


<!-- TabA.vue -->
<script setup>
import { ref } from 'vue'
const count = ref(0)
</script>

<template>
  <h2>Tab A</h2>
  <button @click="count++">Clicked {{ count }} times</button>
</template>
```

5. `<slot>`

Let parent send custom content. Like a gift box where the parent puts anything inside.

```
<!-- Box.vue -->
<template>
  <div style="border: 1px solid gray; padding: 1rem;">
    <slot />
  </div>
</template>
```

```
<!-- App.vue -->
<script setup>
import Box from './Box.vue'
</script>

<template>
  <Box>
    <p>This is some content passed into the Box!</p>
  </Box>
</template>
```

6. `<teleport>`

Send content to another place in HTML. Example: show a modal or popup on top of everything.

```
<script setup>
import { ref } from 'vue'
const showModal = ref(false)
</script>

<template>
  <button @click="showModal = true">Open Modal</button>

  <teleport to="body">
    <div v-if="showModal" class="modal">
      <p>This modal is rendered outside #app!</p>
      <button @click="showModal = false">Close</button>
    </div>
  </teleport>
</template>

<style>
.modal {
  position: fixed;
  top: 40%;
  left: 40%;
  background: white;
  border: 1px solid black;
  padding: 1rem;
  z-index: 1000;
}
</style>
```

7. `<suspense>`

Wait while loading a component. Imagine showing a "Loading..." spinner while a big page loads.

```
<!-- App.vue -->
<script setup>
import { defineAsyncComponent } from 'vue'

const AsyncComp = defineAsyncComponent(() =>
  new Promise(resolve => {
    setTimeout(() => resolve(import('./BigComponent.vue')), 2000)
  })
)
</script>

<template>
  <Suspense>
    <template #default>
      <AsyncComp />
    </template>

    <template #fallback>
      <p>Loading...</p>
    </template>
  </Suspense>
</template>

<!-- BigComponent.vue -->
<template>
  <h2>This is the lazy-loaded component!</h2>
</template>
```