

Composition API Basics

(<script setup>)

Vue 3 is a progressive framework for building user interfaces with reactive data binding.

```
<script setup>
import { ref, reactive, computed,
onMounted } from 'vue'

// Reactive primitive & object state
const count = ref(0)
const user = reactive({ name: 'Alice',
age: 25 })

// Computed property
const doubleCount = computed(() =>
count.value * 2)

// Method
function increment() {
  count.value++
}

onMounted(() => {
  console.log('Component mounted')
})
</script>

<template>
  <button @click="increment">Count:
  {{ count }} (Double:
  {{ doubleCount }})</button>
</template>
```

Template Directives

- **v-bind**: (shorthand **:**) : Dynamic attribute binding (**:href="url"**, **:class="{ active: isActive }"**)
- **v-on**: (shorthand **@**) : Event listener attachment (**@click="handleClick"**, **@submit.prevent="onSubmit"**)
- **v-model**: Two-way data binding (**<input v-model="text" />**)
- **v-if** / **v-else-if** / **v-else**: Conditional DOM insertion/removal
- **v-show**: Toggles CSS **display: none**

- **v-for**: List rendering (**v-for="(item, index) in items :key="item.id"**)
- **v-slot**: (shorthand **#**) : Named and scoped slot content (**#header**)

Reactive Watchers

watch (Explicit Dependencies)

```
import { watch } from 'vue'

watch(count, (newVal, oldVal) => {
  console.log(`Count changed: ${oldVal}
-> ${newVal}`)
}, { immediate: true, deep: true })
```

watchEffect (Automatic Dependency Tracking)

```
import { watchEffect } from 'vue'

watchEffect(() => {
  console.log(`Current count is
${count.value}`)
})
```

Component Communication

defineProps & defineEmits

```
<script setup>
const props = defineProps({
  title: { type: String, required:
true },
  initialCount: { type: Number, default:
0 },
})
```

```
const emit = defineEmits(['update',
'delete'])
```

```
function handleAction() {
  emit('update', count.value)
}
</script>
```

provide & inject (Dependency Injection)

```
// Parent component
import { provide } from 'vue'
provide('themeKey', 'dark')
```

```
// Child component
import { inject } from 'vue'
const theme = inject('themeKey',
'light')
```

Lifecycle Hooks

- **onBeforeMount** / **onMounted**: Before and after DOM mounting
- **onBeforeUpdate** / **onUpdated**: Before and after reactive DOM re-render
- **onBeforeUnmount** / **onUnmounted**: Before and after component teardown
- **onErrorCaptured**: Captures error propagating from descendants

State Management (Pinia Store)

```
import { defineStore } from 'pinia'
import { ref, computed } from 'vue'

export const useCounterStore =
defineStore('counter', () => {
  const count = ref(0)
  const double = computed(() =>
count.value * 2)
  function increment() { count.value++ }

  return { count, double, increment }
})
```

Built-in Components

- **<Transition name="fade">**: Entry/exit animated transitions
- **<KeepAlive>**: Inactive component caching
- **<Teleport to="body">**: Render into target DOM subtree
- **<Suspense>**: Orchestrates async dependency loading states