

컴포지션 API 기본 (<script setup>)

Vue 3는 선언형 템플릿과 반응형 시스템을 갖춘 프로그레시브 웹 프레임워크입니다.

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

// 기본 반응형 상태
const count = ref(0)
const user = reactive({ name: '홍길동', age: 25 })

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

// 메서드 정의
function increment() {
  count.value++
}

onMounted(() => {
  console.log('컴포넌트가 마운트되었습니다.')
})
</script>

<template>
<button @click="increment">클릭: {{ count }} (2배: {{ doubleCount }})</button>
</template>
```

템플릿 디렉티브 (Directives)

- **v-bind**: (축약 **:**) : HTML 속성에 데이터 바인딩 (**:href="url"**, **:class="{ active: isActive }"**)
- **v-on**: (축약 **@**) : 이벤트 리스너 연결 (**@click="handleClick"**, **@submit.prevent="onSubmit"**)
- **v-model**: 양방향 데이터 바인딩 (**<input v-model="text" />**)
- **v-if / v-else-if / v-else**: 조건부 렌더링 (DOM에서 추가/제거)
- **v-show**: 조건부 표시 (**display: none** 토글)

- **v-for**: 리스트 렌더링 (**v-for="(item, index) in items" :key="item.id"**)
- **v-slot**: (축약 **#**): 슬롯 콘텐츠 전달 (**#header**)

반응형 감시 (Watchers)

watch (명시적 의존성 감시)

```
import { watch } from 'vue'

watch(count, (newVal, oldVal) => {
  console.log(`카운트 변경: ${oldVal} → ${newVal}`)
}, { immediate: true, deep: true })
```

watchEffect (자동 의존성 추적)

```
import { watchEffect } from 'vue'

watchEffect(() => {
  console.log(`현재 count 값은 ${count.value}입니다.`)
})
```

컴포넌트 통신 (Props, Emits, Provide/Inject)

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 (의존성 주입)

```
// 부모 컴포넌트
import { provide } from 'vue'
provide('themeKey', 'dark')
```

```
// 자식 컴포넌트
```

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

라이프사이클 훅 (Lifecycle Hooks)

- **onBeforeMount / onMounted**: DOM 마운트 전/후
- **onBeforeUpdate / onUpdated**: 반응형 상태 변경으로 DOM 재렌더링 전/후
- **onBeforeUnmount / onUnmounted**: 컴포넌트 소멸 전/후
- **onErrorCaptured**: 하위 컴포넌트 에러 포착

핀니아 상태 관리 (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 }
})
```

내장 컴포넌트 (Built-in Components)

- **<Transition name="fade">**: 진입/진출 애니메이션
- **<KeepAlive>**: 비활성 컴포넌트 캐싱
- **<Teleport to="body">**: 지정한 DOM 위치로 렌더링 이동
- **<Suspense>**: 비동기 의존성 대기 UI 처리