

Vue 3 Composition API cheat sheet

ref, reactive, computed and watch, lifecycle hooks, props and emits, defineModel, composables, provide/inject and the script setup macros, for Vue 3.5 with TypeScript.



Vue 3.5

Reactive state

ref and reactive

```
import { ref, reactive, shallowRef, toRefs, readonly } from 'vue'

const count = ref(0) // count.value in script, {{ count }} in template
const user = ref<User | null>(null)
const state = reactive({ items: [] as Item[], filter: '' }) // objects only
const { filter } = toRefs(state) // destructure without losing reactivity
const big = shallowRef(hugeList) // only .value is tracked
const locked = readonly(state)
```

computed

```
const total = computed(() => cart.items.reduce((s, i) => s + i.price, 0))
const fullName = computed({
  get: () => `${first.value} ${last.value}`,
  set: (v) => { [first.value, last.value] = v.split(' ') },
})
```

Watchers

watch and watchEffect

```
watch(query, (now, before) => search(now))
watch([a, () => state.b], ([a, b]) => { /* ... */ })
watch(() => state.filter, run, { immediate: true, once: true })
watch(state, save, { deep: true }) // deep: number limits depth (3.5)
watchEffect(() => console.log(count.value)) // runs now, tracks what it reads

watch(id, async (id) => {
  const controller = new AbortController()
  onWatcherCleanup(() => controller.abort()) // 3.5
  data.value = await fetch(`/api/${id}`, { signal: controller.signal }).then((r) => r.json())
})
const stop = watch(src, cb); stop() // also pause() / resume()
```

Lifecycle

Hooks

```
onBeforeMount(() => {})
onMounted(() => el.value?.focus()) // DOM is ready
onBeforeUpdate(() => {}); onUpdated(() => {})
onBeforeUnmount(() => {}); onUnmounted(() => clearInterval(timer))
onErrorCaptured((err) => false) // stop propagation
onActivated(() => {}); onDeactivated(() => {}) // <KeepAlive>
await nextTick() // after the DOM updates
```

Props and emits

Typed, with defaults (3.5 destructure)

```
<script setup lang="ts">
const { title, size = 'md' } = defineProps<{
  title: string
  size?: 'sm' | 'md' | 'lg'
}>()

const emit = defineEmits<{
  change: [id: number]
  update: [value: string]
}>()
emit('change', 1)
</script>
```

Destructured props stay reactive in 3.5; pass them to watch as a getter: watch(() => size, ...).

defineModel (v-model on components)

Child and parent

```
<!-- Child.vue -->
<script setup lang="ts">
const model = defineModel<string>({ required: true })
const count = defineModel<number>('count', { default: 0 })
const [text, modifiers] = defineModel<string, 'trim'>({
  set: (v) => (modifiers.trim ? v.trim() : v),
})
</script>
<template><input v-model="model"></template>

<!-- Parent.vue -->
<Child v-model="name" v-model:count="n" />
```

script setup macros and template refs

Everything at the top level is in the template

```
<script setup lang="ts">
import { useTemplateRef, useId, useAttrs, useSlots } from 'vue'
defineOptions({ inheritAttrs: false })
const input = useTemplateRef<HTMLInputElement>('input') // 3.5
const id = useId() // SSR-safe id (3.5)
const slots = defineSlots<{ default(props: { item: Item }): any }>()
defineExpose({ focus: () => input.value?.focus() })
</script>
<template><label :for="id">Name</label><input :id="id" ref="input"></template>
```

Composables

useX: state + logic you can reuse

```
// composables/useFetch.ts
import { ref, toValue, watchEffect, type MaybeRefOrGetter } from 'vue'

export function useFetch<T>(url: MaybeRefOrGetter<string>) {
  const data = ref<T | null>(null)
  const error = ref<unknown>(null)
  watchEffect(async () => {
    data.value = null; error.value = null
    try { data.value = await (await fetch(toValue(url))).json() }
    catch (e) { error.value = e }
  })
  return { data, error } // return refs, not reactive()
}

const { data } = useFetch<User[]>(() => `/api/users?page=${page.value}`)
```

provide / inject

Typed with an InjectionKey

```
// keys.ts
export const ThemeKey: InjectionKey<Ref<'light' | 'dark'>> =
  Symbol('theme')

// Ancestor
const theme = ref<'light' | 'dark'>('light')
provide(ThemeKey, theme) // provide
readonly(theme) to block writes
app.provide('apiBase', '/api') // app-wide

// Any descendant
const theme = inject(ThemeKey) // Ref | undefined
const base = inject('apiBase', '/api') // with default
```

Checked 2026-10-15 against: <https://vuejs.org/api/reactivity-core.html> · <https://vuejs.org/api/sfc-script-setup.html> · <https://vuejs.org/guide/components/v-model.html> · <https://vuejs.org/guide/reusability/composables.html>

Online version with copy buttons: laravel-vuejs.space/cheatsheets/vue-composition-api · More guides and the monthly newsletter at laravel-vuejs.space