Vue.js Cheat Sheet

Vue 3.x · Composition API + `<script setup>` · verified against **vuejs.org** · ecosystem: router.vuejs.org · pinia.vuejs.org · vitest.dev

CREATE & SFC SKELETON

```
# scaffold on Vite
npm create vue@latest
npm i && npm run dev
```

```
<script setup>
import { ref } from 'vue'
const count = ref(0)
</script>

<template>
  <button @click="count++">{{ count }}</button>
</template>

<style scoped>
button { font-weight: 700 }
</style>
```

Top-level bindings in `<script setup>` are auto-exposed to the template. One root `<style scoped>` per file.

REACTIVITY

```
const n = ref(0)          // n.value in JS
const s = reactive({ a: 1 })   // object only
const d = computed(() => n.value * 2)
const ro = readonly(s)
const { a } = toRefs(s)   // keep reactive
const a2 = toRef(s, 'a')
nextTick(() => {/* DOM updated */})
```

ref: *any value, deep.* **reactive**: *proxy, no destructure, keep the same object. Prefer `ref`.*

```
// advanced
shallowRef() triggerRef() customRef()
shallowReactive() toRaw() markRaw()
effectScope(() => { /* grouped effects */ })
```

TEMPLATE SYNTAX & CLASS/STYLE

```
{{ expr }}              // text interpolation
<span v-html="html">  // raw HTML (XSS!)
:id="x"  :[dynAttr]="v"  // v-bind (+ dynamic arg)
v-bind="objOfAttrs"
@click="fn"  @click="fn($event, id)"  // v-on
@keyup.enter  @submit.prevent  @click.stop.self
:class="{ active: isOn, 'is-err': err }"  // obj
:class="[base, isOn ? on : '']"           // array
:style="{ color, fontSize: size + 'px' }"
```

Shorthands `:` / `@` / `#`. Modifiers `.prevent .stop .self .once .passive`; keys `.enter .esc .exact`. Static + bound `class` merge; on a component they fall through to its root.

DIRECTIVES

```
v-if / v-else-if / v-else   // remove from DOM
v-show="cond"               // toggles display
<li v-for="(x, i) in items" :key="x.id">
v-for="(v, k) in obj"   v-for="n in 10"
<template v-for=…>      // group w/o wrapper
v-once   v-memo="[dep]"  v-pre   v-cloak
```

Always `:key` from data (not index). Don't put `v-if` and `v-for` on the same element.

V-MODEL (FORMS)

```
<input v-model="text">          // :value + @input
<input v-model.number.trim.lazy>
<input type="checkbox" v-model="checked">
<input type="checkbox" v-model="list" value="a">
<input type="checkbox" v-model="ok"
  true-value="yes" false-value="no">
<select v-model="picked" multiple>…</select>
```

*For real validation use **VeeValidate** / schema validation (linked in the reference).*

WATCHERS

```
watch(src, (val, old, onCleanup) => {}, {
  deep: true, immediate: true, once: true,
  flush: 'post'   // 'pre'(def) | 'post' | 'sync'
})
watch([a, () => b.x], ([a, bx]) => {})
const stop = watchEffect(() => { use(a.value) })
onWatcherCleanup(() => controller.abort())
```

Create watchers synchronously in `setup`. Prefer `computed` for pure derived data.

LIFECYCLE HOOKS

```
onBeforeMount  onMounted
onBeforeUpdate onUpdated
onBeforeUnmount onUnmounted
onErrorCaptured  onRenderTracked / Triggered
onActivated / onDeactivated   // <KeepAlive>
onServerPrefetch              // SSR
```

Register synchronously during `setup`. Options API: `created`, `mounted`, `unmounted`, …

COMPONENT API (<SCRIPT SETUP>)

```
const props = defineProps<{ id: number; title?: string }>()
// or runtime + validation:
defineProps({
  id: { type: Number, required: true },
  tags: { type: Array, default: () => [] },
  size: { type: String, validator: v => v > 0 } })
const emit = defineEmits<{ change: [id: number] }>()
const model = defineModel<string>()   // v-model
defineExpose({ focus })
const root = useTemplateRef('root')
useAttrs(); useSlots()
```

Props are one-way / read-only. 3.5: reactive props destructure with defaults.

COMPONENT EVENTS & V-MODEL

```
emit('change', id)
<Child @change="onChange" />

// custom v-model (3.4+)
const title = defineModel('title')
<Child v-model:title="t" />
<Child v-model="a" v-model:b="b" />   // multiple
const [name, mods] = defineModel({ set(v){…} })
```

Pre-3.4: `props.modelValue` + `emit('update:modelValue', v)`.

SLOTS

```
// child
<slot>fallback</slot>
<slot name="header" :count="n" />

// parent
<Card>
  <template #header="{ count }">{{ count }}</template>
  <template #default>body</template>
</Card>
<template #[dynamicName]>…</template>
```

`$slots.x` to test presence. Scoped slot = child passes data up to the slot template.

PROVIDE / INJECT

```
// ancestor
provide('key', readonly(state))
app.provide('key', value)          // app-level

// descendant
const v = inject('key', defaultVal)

// typed
const K = Symbol() as InjectionKey<State>
```

*Skips prop drilling. For app-wide mutable state prefer a **Pinia** store.*

REGISTRATION & FALLTHROUGH ATTRS

```
import Foo from './Foo.vue'   // local (preferred)
app.component('Foo', Foo)     // global

defineOptions({ inheritAttrs: false })
<div v-bind="$attrs">          // place them yourself
const attrs = useAttrs()
```

Multi-word `PascalCase` names. Non-prop attrs, `class` / `style` and listeners auto-apply to the single root; multi-root needs explicit `v-bind="$attrs"`.

BUILT-IN COMPONENTS

```
<Transition name="fade" mode="out-in">…
  // .v-enter-from/-active/-to, .v-leave-…
<TransitionGroup tag="ul" name="list">   // :key req
<KeepAlive :include="/Home/" :max="10">…
<Teleport to="body" :disabled="isMobile">…
<Suspense>
  <template #default><AsyncCmp /></template>
  <template #fallback>Loading…</template>
</Suspense>
```

`<component :is="cmp">` for dynamic components. `<Suspense>` is experimental.

ASYNC COMPONENTS

```
const Chart = defineAsyncComponent(() =>
  import('./Chart.vue'))

defineAsyncComponent({
  loader: () => import('./Chart.vue'),
  loadingComponent, errorComponent,
  delay: 200, timeout: 8000
})
```

Pairs with route-level code splitting and `<Suspense>`.

VUE ROUTER

```
import { createRouter, createWebHistory } from 'vue-router'
const router = createRouter({
  history: createWebHistory(),
  routes: [
    { path: '/', component: Home },
    { path: '/u/:id', component: User, props: true },
    { path: '/settings', children: [ … ] },   // nested
    { path: '/:pathMatch(.*)*', component: NotFound },
    { path: '/lazy', component: () => import('./Lazy.vue') } ]
})
app.use(router)
<RouterLink to="/u/1">  <RouterView />
router.beforeEach((to, from) => {/* false | route */})
const route = useRoute(); const r = useRouter()
r.push({ name: 'user', params: { id } })
```

PINIA

```
import { defineStore } from 'pinia'
// setup store
export const useCart = defineStore('cart', () => {
  const items = ref([])
  const count = computed(() => items.value.length)
  const add = p => items.value.push(p)
  return { items, count, add }
})
// option store
defineStore('cart', {
  state: () => ({ items: [] }),
  getters: { count: s => s.items.length },
  actions: { add(p) { this.items.push(p) } } })

const cart = useCart()
const { items, count } = storeToRefs(cart)   // keep reactive
cart.$patch({…}) · $reset() · $subscribe(cb) · $onAction(cb)
```

TESTING — VITEST + TEST UTILS

```
import { mount } from '@vue/test-utils'
import { expect, test } from 'vitest'

test('increments', async () => {
  const w = mount(Counter, { props: { start: 1 } })
  await w.get('button').trigger('click')
  expect(w.text()).toContain('2')
  expect(w.emitted('change')[0]).toEqual([2])
})
```

`shallowMount`, `findComponent`, `setValue`, `flushPromises()`, global `plugins: [router, pinia]`. E2E: Cypress / Playwright.

SSR & HYDRATION

```
// server
const app = createSSRApp(App)
const html = await renderToString(app)
// client — attaches to existing DOM
createSSRApp(App).mount('#app')
```

*Use factory functions to avoid cross-request state. Mismatch → `data-allow-mismatch`. Full solution: **Nuxt**; content sites: **VitePress**.*

CUSTOM DIRECTIVES & PLUGINS

```
const vFocus = {
  mounted: (el, binding) => el.focus()
  // updated, beforeUnmount; binding.value/.arg/.modifiers
}
<input v-focus>

const plugin = {
  install(app, options) {
    app.component(…); app.directive(…)
    app.provide(…); app.config.globalProperties.$t = …
  }
}
app.use(plugin, { locale: 'en' })
```

*Prefer **composables** over mixins for reusable logic.*

COMPOSABLES

```
export function useMouse() {
  const x = ref(0), y = ref(0)
  const move = e => { x.value = e.pageX; y.value = e.pageY }
  onMounted(() => addEventListener('mousemove', move))
  onUnmounted(() => removeEventListener('mousemove', move))
  return { x, y }
}
```

Name it `use*`; call synchronously in `setup`; accept `MaybeRefOrGetter` and read via `toValue()`.

RENDER FUNCTION & JSX

```
import { h } from 'vue'
h('div', { class: 'box', onClick }, [h('span', text)])

// functional component
const Hi = (props) => h('p', 'Hi ' + props.name)
```

Compiler-informed VDOM: static hoisting + patch flags. Reach for `h()` only for highly dynamic structure.

VUE + TYPESCRIPT

```
const n = ref<number>(0)
defineProps<{ id: number }>()
withDefaults(definePropsProps>(), { size: 40 })
defineEmits<{ submit: [payload: Form] }>()
const el = useTemplateRef<HTMLInputElement>('el')
const K = Symbol() as InjectionKey<Store>

// Options API
export default defineComponent({ props: { x: Number as
PropType<X> } })
```

Scaffold with `create-vue` (TS) + `vue-tsc`. See the TypeScript Reference for the language.

PERFORMANCE

```
v-memo="[a, b]"        // skip subtree unless deps change
shallowRef / shallowReactive   // large immutable data
<KeepAlive>             // cache toggled views
defineAsyncComponent + route-level import()
computed()              // cache derived data
```

Prod flags: `__VUE_OPTIONS_API__`, `__VUE_PROD_DEVTOOLS__`. Virtualize long lists; keep `:key` stable.

TOOLING & DEVTOOLS

```
npm create vue@latest        // create-vue on Vite
npm run dev / build / preview
vue-tsc --noEmit             // type-check SFCs
eslint-plugin-vue + Prettier
```

Volar *(Vue Language Tools) for the editor.* **Vue DevTools**: *browser ext (v7), standalone app, or Vite plugin — tree, state, timeline, router/Pinia.*

SECURITY

```
{{ userInput }}     // auto HTML-escaped — safe
v-html="userInput"  // XSS — never with untrusted input
:href="userUrl"     // validate scheme (no javascript:)
```

Never compile a template from user input. See the Security & Accessibility page + `xref:web/cors.adoc`.

OPTIONS ⇄ COMPOSITION

```
data()              -> ref() / reactive()
computed: {}        -> computed()
methods: {}         -> plain functions
watch: {}           -> watch() / watchEffect()
mounted()           -> onMounted()
props / emits       -> defineProps / defineEmits
provide/inject      -> provide() / inject()
mixins              -> composables (use*)
```