

# REFERÊNCIA RÁPIDA DO VUE.JS

Templates, reatividade, componentes, Composition API, router

## Sintaxe de Template

### Texto e Expressões

```
<span>{{ message }}</span>
<span>{{ count + 1 }}</span>
<span>{{ ok ? 'Yes' : 'No' }}</span>
<span v-html="rawHtml"></span>
```

### Diretivas

<b>{{ expr }}</b>	Interpolação de texto
<b>v-bind:attr / :attr</b>	Vincular atributo à expressão
<b>v-on:event / @event</b>	Adicionar ouvinte de evento
<b>v-model</b>	Vinculação bidirecional (formulários)
<b>v-if / v-else-if / v-else</b>	Renderização condicional
<b>v-show</b>	Alternar CSS de exibição (permanece no DOM)
<b>v-for</b>	Renderização de lista
<b>v-slot / #name</b>	Conteúdo de slot nomeado

### Vinculação de Atributos

```

<div :class="{ active: isActive }"></div>
<div :style="{ color: textColor }"></div>
<button :disabled="isLoading">Submit</button>
```

## Reatividade

### ref (Primitivos)

```
import { ref } from 'vue'

const count = ref(0)
console.log(count.value) // 0
count.value++ // reactive update
```

### reactive (Objetos)

```
import { reactive } from 'vue'

const state = reactive({ count: 0, name: 'Vue' })
state.count++ // no .value needed
```

### ref vs reactive

- ref()** Qualquer tipo; acessar via `.value` no script
- reactive()** Apenas objetos/arrays; acesso direto às propriedades
- Template** Ambos se desempacotam automaticamente (sem `.value`)

**Destructure** `reactive`` perde reatividade; usar `toRefs()`

## Computed e Watchers

### Propriedades Computed

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

const items = ref([1, 2, 3, 4, 5])
const evenItems = computed(() =>
  items.value.filter(n => n % 2 === 0))
```

Valores computed são armazenados em cache e reavaliados apenas quando as dependências mudam

### Watchers

```
import { ref, watch, watchEffect } from 'vue'

const query = ref('')

// Watch specific source
watch(query, (newVal, oldVal) => {
  console.log('Changed: ${oldVal} -> ${newVal}')
})

// Auto-track dependencies
watchEffect(() => {
  console.log('Query is: ${query.value}')
})
```

### Opções de Watch

<b>immediate: true</b>	Executar callback imediatamente ao criar
<b>deep: true</b>	Observar objetos aninhados profundamente
<b>flush: 'post'</b>	Executar após atualização do DOM
<b>once: true</b>	Disparar apenas uma vez e parar

## Componentes

### Componente de Arquivo Único (SFC)

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

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

<style scoped>
button { font-size: 1.2em; }
</style>
```

### Registrando Componentes

```
<!-- Auto-imported with <script setup> -->
<script setup>
import MyButton from './MyButton.vue'
</script>

<template>
<MyButton label="Click me" />
</template>
```

### Blocos SFC

<b>&lt;script setup&gt;</b>	Composition API (recomendado)
<b>&lt;template&gt;</b>	Template HTML
<b>&lt;style scoped&gt;</b>	CSS com escopo do componente
<b>&lt;style module&gt;</b>	CSS Modules (objeto \$style)

## Props e Eventos

### Definindo Props

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

### Emittindo Eventos

```
<script setup>
const emit = defineEmits(['update', 'delete'])

function handleClick() {
  emit('update', { id: 1, value: 'new' })
}
</script>
```

### Uso no Pai

```
<ChildComponent
  :title="pageTitle"
  :count="total"
  @update="handleUpdate"
  @delete="handleDelete"
/>
```

### v-model em Componentes

```
<!-- Parent -->
<CustomInput v-model="search" />

<!-- CustomInput.vue -->
<script setup>
const model = defineModel()
</script>
<template>
<input :value="model" @input="model = $event.target.value" />
</template>
```

## Slots

### Slot Padrão

```
<!-- Card.vue -->
<template>
<div class="card">
  <slot>Fallback content</slot>
</div>
</template>
```

```
<!-- Usage -->
<Card><p>Custom content here</p></Card>
```

### Slots Nomeados

```
<!-- Layout.vue -->
<template>
<header><slot name="header" /></header>
<main><slot /></main>
<footer><slot name="footer" /></footer>
</template>

<!-- Usage -->
<Layout>
<template #header><h1>Title</h1></template>
<p>Main content</p>
<template #footer><span>Footer</span></template>
</Layout>
```

### Slots com Escopo

```
<!-- List.vue -->
<ul>
<li v-for="item in items" :key="item.id">
  <slot :item="item" />
</li>
</ul>

<!-- Usage -->
<List items="todos">
<template #default="{ item }">
<span>{{ item.text }}</span>
</template>
</List>
```

## Composition API

### Função Composable

```
// useMouse.js
import { ref, onMounted, onUnmounted } from 'vue'

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

### Usando Composables

```
<script setup>
import { useMouse } from './useMouse'

const { x, y } = useMouse()
</script>
<template>
<p>Mouse: {{ x }}, {{ y }}</p>
</template>
```

### provide / inject

```
// Parent
import { provide, ref } from 'vue'
const theme = ref('dark')
provide('theme', theme)

// Descendant (any depth)
import { inject } from 'vue'
const theme = inject('theme', 'light') // default
```

## Router (Vue Router)

### Definições de Rota

```
import { createRouter, createWebHistory } from 'vue-router'

const router = createRouter({
  history: createWebHistory(),
  routes: [
    { path: '/', component: Home },
    { path: '/about', component: About },
    { path: '/user/:id', component: User },
  ]
})
```

### Navegação no Template

```
<router-link to="/Home">router-link</router-link>
<router-link :to="{ name: 'user', params: { id: 1 } }">
  User 1
</router-link>
<router-view />
```

### Navegação Programática

```
import { useRouter, useRoute } from 'vue-router'

const router = useRouter()
const route = useRoute()

router.push('/about')
router.push({ name: 'user', params: { id: 1 } })
console.log(route.params.id)
```

## Recursos de Rota

<b>/user/:id</b>	Segmento dinâmico ( <code>^ route.params.id</code> )
<b>(name: 'user')</b>	Rota nomeada para navegação programática
<b>children: [...]</b>	Rotas aninhadas
<b>beforeEnter</b>	Guarda de navegação por rota
<b>meta: { auth: true }</b>	Metadados personalizados para guardas
<b>redirect: '/new-path'</b>	Redirecionamento de rota

## Hooks de Ciclo de Vida

### Ordem dos Hooks

<b>onBeforeMount</b>	Antes da renderização inicial do DOM
<b>onMounted</b>	DOM pronto (buscar dados, adicionar ouvintes)
<b>onBeforeUpdate</b>	Antes do estado reativo re-renderizar o DOM
<b>onUpdated</b>	Após re-renderização do DOM
<b>onBeforeUnmount</b>	Antes do componente ser destruído
<b>onUnmounted</b>	Limpeza (remover ouvintes, timers)

### Uso

```
<script setup>
import { onMounted, onUnmounted } from 'vue'

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

onUnmounted(() => {
  console.log('Cleanup here')
})
</script>
```

## Listas e Condicionais

### v-for

```
<li v-for="item in items" :key="item.id">
  {{ item.name }}
</li>

<li v-for="(item, index) in items" :key="item.id">
  {{ index }}: {{ item.name }}
</li>

<div v-for="(val, key) in obj" :key="key">
  {{ key }}: {{ val }}
</div>
```

Sempre usar `key` com `v-for` para atualizações eficientes do DOM

### v-if vs v-show

<b>v-if</b>	Renderizar condicionalmente (adicionar/remover do DOM)
<b>v-else-if</b>	Cadeia else-if
<b>v-else</b>	Ramo de fallback
<b>v-show</b>	Alternar <code>display: none</code> (permanece no DOM)

Usar `v-show` para alternâncias frequentes, `v-if` para mudanças raras

### Exemplo de Condicionais

```
<div v-if="status === 'loading'">Loading...</div>
<div v-else-if="status === 'error'">Error!</div>
<div v-else>{{ data }}</div>
```

## Manipulação de Formulários

### Básico de v-model

```
<input v-model="text" />
<textarea v-model="message"></textarea>
<input type="checkbox" v-model="checked" />
<select v-model="selected">
  <option value="a">A</option>
  <option value="b">B</option>
</select>
```

### Modificadores de v-model

<b>v-model.lazy</b>	Sincronizar em <code>'change'</code> em vez de <code>'input'</code>
<b>v-model.number</b>	Auto-converter para número
<b>v-model.trim</b>	Auto-remover espaços em branco

### Modificadores de Evento

<b>@click.prevent</b>	Chamar <code>preventDefault()</code>
<b>@click.stop</b>	Chamar <code>stopPropagation()</code>
<b>@click.once</b>	Disparar no máximo uma vez
<b>@keyup.enter</b>	Apenas na tecla Enter
<b>@submit.prevent</b>	Prevenir envio padrão do formulário