RIFERIMENTO RAPIDO VUE.JS

Template, reattività, componenti, Composition API, router

Sintassi dei Template

Testo ed Espressioni

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

Direttive

`{{ expr }}`	Interpolazione di testo
`v-bind:attr / :attr`	Lega un attributo a un'espressione
`v-on:event / @event`	Aggiunge un listener di eventi
`v-model`	Binding bidirezionale (form)
`v-if / v-else-if / v-else`	Rendering condizionale
`v-show`	Alterna il CSS display (rimane nel DOM)
`v-for`	Rendering di liste
`v-slot / #name`	Contenuto slot con nome

Binding degli Attributi

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

Reattività

ref (Primitivi)

```
import { ref } from 'vue'

const count = ref(0)
console.log(count.value)  // 0
count.value++             // aggiornamento reattivo
```

reactive (Oggetti)

```
import { reactive } from 'vue'

const state = reactive({ count: 0, name: 'Vue' })
state.count++  // nessun .value necessario
```

ref vs reactive

`ref()`	Qualsiasi tipo; accesso tramite `.value` nello script
`reactive()`	Solo oggetti/array; accesso diretto alle proprietà
`Template`	Entrambi si decomprimono automaticamente (nessun `.value`)
`Destructure`	`reactive` perde la reattività; usa `toRefs()`

Computed e Watcher

Proprietà Computed

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

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

I valori computed sono in cache e si ricalcolano solo quando le dipendenze cambiano

Watcher

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

const query = ref('')

// Osserva una sorgente specifica
watch(query, (newVal, oldVal) => {
  console.log(`Changed: ${oldVal} → ${newVal}`)
})

// Traccia automaticamente le dipendenze
watchEffect(() => {
  console.log(`Query is: ${query.value}`)
})
```

Opzioni Watch

`immediate: true`	Esegui il callback immediatamente alla creazione
`deep: true`	Osservazione profonda di oggetti annidati
`flush: 'post'`	Esegui dopo l'aggiornamento del DOM
`once: true`	Si attiva una sola volta poi si ferma

Componenti

Single-File Component (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>
```

Registrazione dei Componenti

```
<!-- Auto-importato con <script setup> -->
<script setup>
import MyButton from './MyButton.vue'
</script>

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

Blocchi SFC

`<script setup>`	Composition API (consigliato)
`<template>`	Template HTML
`<style scoped>`	CSS con scope al componente
`<style module>`	CSS Modules (oggetto $style)

Props ed Eventi

Definizione Props

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

Emissione di Eventi

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

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

Utilizzo nel Componente Padre

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

v-model sui Componenti

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

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

Slot

Slot Predefinito

```
<!-- Card.vue -->
<template>
  <div class="card">
    <slot>Contenuto di fallback</slot>
  </div>
</template>

<!-- Utilizzo -->
<Card><p>Contenuto personalizzato</p></Card>
```

Slot con Nome

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

<!-- Utilizzo -->
<Layout>
  <template #header><h1>Titolo</h1></template>
  <p>Contenuto principale</p>
  <template #footer><span>Footer</span></template>
</Layout>
```

Slot con Scope

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

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

Composition API

Funzione 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 }
}
```

Utilizzo dei Composable

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

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

provide / inject

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

// Discendente (qualsiasi profondità)
import { inject } from 'vue'
const theme = inject('theme', 'light') // valore predefinito
```

Router (Vue Router)

Definizione Rotte

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

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

Navigazione nel Template

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

Navigazione Programmatica

```
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)
```

Funzionalità delle Rotte

`/user/:id`	Segmento dinamico (`route.params.id`)
`name: 'user'`	Rotta con nome per navigazione programmatica
`children: [...]`	Rotte annidate
`beforeEnter`	Guardia di navigazione per singola rotta
`meta: { auth: true }`	Metadati personalizzati per le guardie
`redirect: '/new-path'`	Reindirizzamento della rotta

Hook del Ciclo di Vita

Ordine degli Hook

`onBeforeMount`	Prima del rendering iniziale del DOM
`onMounted`	Il DOM è pronto (fetch dati, aggiunta listener)
`onBeforeUpdate`	Prima che lo stato reattivo ri-renderizzi il DOM
`onUpdated`	Dopo il re-rendering del DOM
`onBeforeUnmount`	Prima che il componente venga distrutto
`onUnmounted`	Pulizia (rimozione listener, timer)

Utilizzo

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

onMounted(() => {
  console.log('Componente montato')
})

onUnmounted(() => {
  console.log('Pulizia qui')
})
</script>
```

Liste e Condizionali

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>
```

Usa sempre :key con v-for per aggiornamenti efficienti del DOM

v-if vs v-show

`v-if`	Rendering condizionale (aggiunge/rimuove dal DOM)
`v-else-if`	Catena else-if
`v-else`	Ramo di fallback
`v-show`	Alterna `display: none` (rimane nel DOM)

Usa v-show per toggle frequenti, v-if per cambiamenti rari

Esempio Condizionali

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

Gestione dei Form

Basi di 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>
```

Modificatori v-model

`v-model.lazy`	Sincronizza su `change` invece che su `input`
`v-model.number`	Converte automaticamente in numero
`v-model.trim`	Rimuove automaticamente gli spazi iniziali/finali

Modificatori di Evento

`@click.prevent`	Chiama `preventDefault()`
`@click.stop`	Chiama `stopPropagation()`
`@click.once`	Si attiva al massimo una volta
`@keyup.enter`	Solo al tasto Invio
`@submit.prevent`	Impedisce l'invio predefinito del form