Guida rapida a Vue.js

Stefanoo94

Contents

Guida rapida a Vue.js

Versione: 1.0
Autore: Stefanoo94

Scopo

Questa guida fornisce i passaggi pratici per iniziare con Vue.js (consigliato Vue 3), creare progetti, usare Composition API e Options API, routing, gestione stato con Pinia, testing, build e deploy. Contiene esempi pratici e comandi utili.

Prerequisiti

- Node.js LTS (consigliato) e npm o yarn
- Git
- Conoscenza base di JavaScript (ES6+), HTML e CSS

- (Opzionale) TypeScript se vuoi usare Vue con tipi

---

1. Creare un nuovo progetto (consigliato: Vite)

Vite è il modo raccomandato per nuovi progetti Vue 3:

```
# con npm
npm create vite@latest nome-app -- --template vue

# oppure con yarn
yarn create vite nome-app --template vue

# entra nella cartella e installa
cd nome-app
npm install
npm run dev
```

Per TypeScript:

```
npm create vite@latest nome-app -- --template vue-ts
```

Alternative legacy: - Vue CLI (solo se serve legacy scaffolding): `npm install -g @vue/cli` e `vue create nome-app`

---

2. Struttura consigliata

Una struttura semplice e scalabile:

```
/src
  /assets
  /components
    BaseButton.vue
  /views
    Home.vue
    About.vue
  /composables    # composable functions (useX)
  /stores         # Pinia stores
  /router
    index.js
  App.vue
  main.js (main.ts)
public/
index.html
package.json
```

---

3. Entry point e bootstrap

Esempio main (Vue 3 + Router + Pinia):

```
// src/main.js
import { createApp } from 'vue'
import { createPinia } from 'pinia'
```

```
import router from './router'
import App from './App.vue'
import './styles.css'

const app = createApp(App)
app.use(createPinia())
app.use(router)
app.mount('#app')
```

---

4. Componenti: Options API vs Composition API

Options API (classico):

```
<template>
  <div>{{ message }}</div>
</template>
<script>
export default {
  data() {
    return { message: 'Ciao Vue!' }
  },
  methods: {
    doSomething() { /* ... */ }
  }
}
</script>
```

Composition API (consigliato per riusabilità e TypeScript):

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

<script setup>
import { ref } from 'vue'

const count = ref(0)
function increment() { count.value++ }
</script>
```

---

5. Reactive, ref e computed

- ref crea un valore reattivo scalare: `const x = ref(0)`
- reactive crea un oggetto reattivo: `const state = reactive({ items: [] })`
- computed per valori derivati:

```
import { computed } from 'vue'
const double = computed(() => count.value * 2)
```

---

6. Props, emits e slots

Props e emits:

```
<script setup>
defineProps({ title: String })
const emit = defineEmits(['close'])
</script>
```

Slots:

```
<!-- Parent.vue -->
<MyCard>
  <template #header>Header</template>
  Contenuto
</MyCard>
```

---

7. Router (vue-router)

Installazione:

```
npm install vue-router@4
```

Esempio base:

```
// src/router/index.js
import { createRouter, createWebHistory } from 'vue-router'
import Home from '../views/Home.vue'
import About from '../views/About.vue'

const routes = [
  { path: '/', component: Home },
  { path: '/about', component: About }
]

const router = createRouter({
  history: createWebHistory(),
  routes
})

export default router
```

Navigazione programmata:

```
import { useRouter } from 'vue-router'
const router = useRouter()
router.push({ name: 'home' })
```

---

8. Stato globale: Pinia (sostituto moderno di Vuex)

Installazione:

```
npm install pinia
```

Esempio store:

```
// src/stores/counter.js
import { defineStore } from 'pinia'
export const useCounterStore = defineStore('counter', {
  state: () => ({ count: 0 }),
  actions: {
    increment() { this.count++ }
  }
})
```

Uso in componente:

```
import { useCounterStore } from '@/stores/counter'
const counter = useCounterStore()
counter.increment()
```

---

9. Comunicazione con API

- Usa fetch o axios:

```
npm install axios
```

Esempio service:

```
// src/services/api.js
import axios from 'axios'
export const api = axios.create({ baseURL: import.meta.env.VITE_API_URL || '/api' })
```

Gestisci token con interceptors:

```
api.interceptors.request.use(config => {
  const token = localStorage.getItem('token')
  if (token) config.headers.Authorization = `Bearer ${token}`
  return config
})
```

---

10. Forms e validazione

- VeeValidate o vuelidate per validazioni complesse. Esempio con VeeValidate:

```
npm install vee-validate yup @vee-validate/rules
```

---

11. Testing

- Unit e integrazione: Vitest + Vue Testing Library

```
npm install -D vitest @testing-library/vue @testing-library/jest-dom
```

Esempio test:

```
import { render } from '@testing-library/vue'
import Counter from '../components/Counter.vue'

test('renders counter', () => {
```

```
  const { getByText } = render(Counter)
  getByText(/Count/i)
})
```

---

12. Linting e formatting

- ESLint + Prettier + plugin Vue:

```
npm install -D eslint prettier eslint-plugin-vue
npx eslint --init
```

- Husky + lint-staged per pre-commit:

```
npm install -D husky lint-staged
npx husky install
# package.json add: "lint-staged": { "*.vue": "eslint --fix" }
```

---

13. Internationalization (i18n)

- Usa vue-i18n:

```
npm install vue-i18n
```

Setup e uso semplice con composable `useI18n`.

---

14. SSR e SSG (Nuxt)

- Per SSR/SSG/Meta, considera Nuxt 3:
  - Nuxt gestisce routing, store, rendering lato server, e molti plugin.
  - Comando per iniziare: `npx nuxi init nome-app && npm install && npm run dev`

---

15. Build e deploy

- Build:

```
npm run build
```

- Deploy:
  - Vercel: collega il repo e deploy automatico
  - Netlify: collega repo o upload build
  - GitHub Pages (SPA): usare gh-pages o configurare redirect corretto
  - Docker: creare immagine e deploy su servizi containerizzati

---

16. Performance e best practices

- Code splitting via router lazy loading:

```
const About = () => import('../views/About.vue')
```

- Ottimizza immagini, usa lazy loading, evita re-render inutili.

- Usa devtools Vue e performance profiling.

---

17. Migrazione e upgrade

- Segui la guida ufficiale di aggiornamento Vue (major changes)
- Usa `vue-migration-helper` e tool di lint per deprecazioni

---

18. Risorse utili

- Docs Vue: https://vuejs.org
- Vite: https://vitejs.dev
- Vue Router: https://router.vuejs.org
- Pinia: https://pinia.vuejs.org
- Nuxt: https://nuxt.com
- Vue Test Utils / Testing Library: https://testing-library.com

---

Esempio minimo App.vue

```
<template>
  <div>
    <h1>Benvenuto in Vue</h1>
    <Counter />
  </div>
</template>

<script setup>
import Counter from './components/Counter.vue'
</script>
```

Checklist prima del deploy

- ☐ Variabili d'ambiente configurate (VITE_*)
- ☐ Lint e test OK
- ☐ Build verificata localmente
- ☐ File sensibili esclusi (.env)

Vuoi che generi anche il PDF automaticamente (comando pandoc pronto) e lo aggiunga al repository come fatto per le guide precedenti? Posso anche creare guida-vue.pdf, aggiungere il commit e pushare i file; dimmi se preferisci usare XeLaTeX o wkhtmltopdf per la generazione del PDF.