■ Guia Completo Vue.js - Manual de Bolso

Básico

■ **data**: Armazena os dados reativos do componente.

➡■ Contexto: Contexto: Criar um campo de nome que aparece na tela.

■ Exemplo:

Exemplo: data() { return { nome: 'Maria' } }

■ **methods**: Funções do componente que podem ser chamadas em eventos ou no template.

➡■ Contexto: Contexto: Botão que mostra uma mensagem ao clicar.

■ Exemplo:

Exemplo: methods: { ola() { alert('Olá Vue!') } }

■ **computed**: Propriedades derivadas reativas baseadas em outras.

➡■ Contexto: Contexto: Mostrar nome completo a partir de nome e sobrenome.

■ Exemplo:

Exemplo: computed: { nomeCompleto() { return this.nome + ' ' + this.sobrenome } }

■ **watch**: Observa mudanças em propriedades reativas e executa lógica quando mudam.

➡■ Contexto: Contexto: Buscar dados da API sempre que 'id' mudar.

■ Exemplo:

Exemplo: watch: { id(novo) { this.buscarUsuario(novo) } }

Renderização

■ **v-if / v-else**: Renderiza condicionalmente elementos.

➡■ Contexto: Contexto: Mostrar botão 'Login' se não estiver logado e 'Logout' se estiver.

■ Exemplo:

Exemplo: LogoutLogin

■ **v-for**: Renderiza listas com base em arrays ou objetos.

➡■ Contexto: Contexto: Mostrar lista de tarefas.

■ Exemplo:

Exemplo: {{ tarefa.nome }}

Componentes

■ **props**: Permite passar dados de um componente pai para um filho.

➡■ Contexto: Contexto: Componente 'Card' recebe título via prop.

■ Exemplo:

Exemplo: props: ['titulo']

■ **emit**: Permite que o filho envie eventos para o pai.

➡■ Contexto: Contexto: Botão no filho avisa ao pai que foi clicado.

■ Exemplo:

Exemplo: this.$emit('clicado')

Reatividade

■ **ref**: Cria valores reativos no Composition API.

➡■ Contexto: Contexto: Contador simples.

■ Exemplo:

Exemplo: const contador = ref(0)

■ **reactive**: Cria objetos reativos.

➡■ Contexto: Contexto: Usuário com várias propriedades.

■ Exemplo:

Exemplo: const usuario = reactive({ nome: 'João', idade: 30 })

Ciclo de Vida

■ **mounted**: Executa após o componente estar montado.

➡■ Contexto: Contexto: Carregar dados da API ao iniciar.

■ Exemplo:

Exemplo: mounted() { this.fetchData() }

■ **updated**: Executa após atualização do DOM.

➡■ Contexto: Contexto: Mostrar mensagem quando algo mudar.

■ Exemplo:

Exemplo: updated() { console.log('Componente atualizado') }

Integrações

■ **Axios**: Cliente HTTP para consumir APIs.

➡■ Contexto: Contexto: Buscar lista de usuários.

■ Exemplo:

Exemplo: axios.get('/api/usuarios').then(r => this.usuarios = r.data)

■ **Vue Router**: Gerencia rotas na aplicação.

➡■ Contexto: Contexto: Ter páginas Home e About.

■ Exemplo:

Exemplo: { path: '/about', component: About }

■ Checklist de Prática

- Usei v-for para renderizar listas?
- Treinei watch para reagir a mudanças de estado?
- Criei um componente filho recebendo props?
- Implementei eventos com emit?
- Experimentei lifecycle hooks (mounted, updated)?
- Consumi uma API real com axios?
- Configurei rotas básicas com Vue Router?
- Testei reactive e ref no Composition API?