Mapa Mental do bloco SCRIPT no Vue.js

Este documento funciona como um guia rápido (índice/mindmap) para consulta no dia a dia. Mostra os principais elementos que podem ser usados dentro do bloco 'script' em componentes Vue, tanto na Options API quanto na Composition API.

1) Options API (forma clássica)

- **name** → nome do componente
- **components** → registrar componentes filhos
- **props** → dados recebidos do componente pai
- **data()** → estado reativo local
- **methods** → funções do componente
- **computed** → propriedades derivadas, cacheadas
- **watch** → observar mudanças em variáveis
- **emits** → eventos que o componente pode emitir
- **lifecycle hooks** → funções chamadas em momentos do ciclo de vida (created, mounted, updated, unmounted, etc.)

2) Composition API (forma moderna, Vue 3+)

- **importações da Vue API** → ref, reactive, computed, watch, onMounted, defineProps, defineEmits...
- **setup()** ou **<script setup>** → ponto central da lógica
- **ref()** → valores reativos simples
- **reactive()** → objetos reativos
- **computed()** → valores derivados
- **watch()** → reação a mudanças
- **watchEffect()** → executa sempre que dependências mudam
- **lifecycle hooks** → onMounted, onUpdated, onUnmounted...
- **defineProps()** → recebe dados do pai
- **defineEmits()** → emite eventos

3) Extras Importantes

- **Mixins** → reuso de lógica (Options API)
- **Composables** → funções reutilizáveis (Composition API)
- **provide / inject** → compartilhar estado entre componentes sem props
- **Store (Vuex / Pinia)** → gerenciamento de estado global

Resumo prático do uso do bloco SCRIPT

Se quiser estado → use data() ou ref/reactive
Se quiser lógica → use methods ou funções no setup
Se quiser valores derivados → use computed
Se quiser reagir a mudanças → use watch ou watchEffect
Se quiser executar em momentos específicos → use lifecycle hooks