Composables

Co to je a jak je správně používat

Co to sakra je?

- Composable je funkce, která využívá reaktivitu nebo lifecycle hooks
- Lze ji znovu použít v různých částech aplikace.
- Umožňuje oddělit a znovu použít logiku mezi různými komponentami.
- Pomáhá udržovat kód čistý, modulární a snadno testovatelný.

Jak vytvořit Composable?

- Každá composable funkce by měla začínat předponou "use", např. `useFetchData`.
- měla by vracet reaktivní stav (např. `ref`, `reactive`)
- + metody, které tento stav manipulují.
- Měla by být umístě v samostatném souboru, obvykle ve složce `composables`. (Nuxt.js má tuto složku předdefinovanou)
- pokud používáte TypeScript, je dobré přidat typové anotace pro lepší typovou kontrolu.

Příklad Composable

```
1 import { ref } from 'vue';
2 export function useCounter() {
3   const count = ref(0);
4
5   function increment() {
6     count.value++;
7   }
8
9   function decrement() {
10    count.value--;
11  }
12
13  return {
14    count, increment, decrement
15  };
16 }
```

Jak používat composable?

- composable může být použit v setup funkci komponenty
- composable může být použit v jiném composable
- nesmí být zanořena uvnitř jiných funkcí
- nesmí být volána za await nebo uvnitř callbacků

Příklad použití Composable v komponentě

```
<script setup lang="ts">
import { useCounter } from '@/composables/useCounter';
const { count, increment, decrement } = useCounter();
</script>
<template>
  <div>
    <p>Count: {{ count }}</p>
    <button @click="decrement">-</button>
    <button @click="increment">+</button>
  </div>
</template>
```

Příklad zanoření Composable

```
1 import { useCounter } from '@/composables/useCounter';
2 export function useAdvancedCounter() {
3   const { count, increment, decrement } = useCounter();
4
5   function reset() {
6     count.value = 0;
7   }
8
9   return {
10    count,
11    increment,
12    decrement,
13    reset
14  };
15}
```

Jak si pohlídat správné použití?

- použijeme eslint plugin `eslint-plugin-vue-composable`
- zajistí, že composable budou používány správně, tím že nás mlátí přes prsty, když je použijeme špatně

Rychlé tipy

- i Pinia stores jsou vlastně composables 🍍
- composables by měly být čisté funkce bez vedlejších efektů
- `vueuse` je skvělá knihovna plná užitečných composables (https://vueuse.org/)
  - `useLocalStorage`
  - `useEventBus`

Typescript tip na závěr

Pokud má composable parametry je dobrá řešit reaktivní i nereaktivní vstupy pomocí ➡️ `MaybeRef` ⬅️

```
1 import { MaybeRef, ref } from 'vue';
2 export function useExample(param: MaybeRef<string>) {
3   // ref nic nezmění pokud je již reaktivní
4   const value = ref(param);
5   // ...
6 }
```

Díky za pozornost!

Prezentaci pro vás s radostí připravil z více než 90% Copilot, protože mně by to trvalo moc dlouho 🤣