Vite boilerplate

Ora è giunto il momento di rendere le nostre applicazioni più strutturate, per farlo adopereremo il boilerplate ufficiale(https://vuejs.org/guide/quick-start.html) :

1. Lancia npm create vue@latest nel tuo terminale
2. Scegli il nome del progetto
3. Premi invio ai 2 passaggi successivi senza selezionare opzioni(per ora)
4. Scegli se usare un blank template o lo starter
5. Lancia **npm i** nel progetto
6. Lancia **npm run dev** per avviare l’anteprima

I componenti I

La prima sezione è <script>, che rappresenta la parte logica del componente.

Qui vengono definiti i dati, le funzioni e le variabili reattive necessarie al funzionamento. Con l'approccio moderno, spesso si utilizza la sintassi <script setup>, che semplifica l'uso delle API di Vue e riduce la necessità di codice ridondante.

È in questa sezione che si importano altri componenti o librerie e si gestisce la logica di interazione.

I componenti II

La seconda sezione è <template>, dedicata alla struttura visiva del componente.

In questa parte si descrive ciò che l’utente vedrà a schermo, attraverso una sintassi simile all’HTML.

Qui i dati dichiarati nello script vengono collegati direttamente all’interfaccia, rendendo possibile la reattività: ogni variazione nei dati produce automaticamente un aggiornamento nella vista.

I componenti III

la terza sezione è <style>, che si occupa della definizione dell'aspetto grafico.

Qui si possono scrivere regole CSS per personalizzare la resa visiva del componente. Spesso si utilizza l'attributo scoped, in modo che gli stili rimangano isolati e non vadano a influenzare altri componenti dell'applicazione.

Un file di componente Vue ha generalmente estensione .vue ed è composto da tre sezioni principali, racchiuse in tag specifici.

ref()

La funzione ref() è lo strumento fondamentale che consente di creare un dato reattivo, cioè una variabile che, quando modificata, provoca automaticamente l'aggiornamento dell'interfaccia utente.

Dal punto di vista pratico, quando si invoca ref(), Vue restituisce un oggetto speciale che contiene una proprietà chiamata .value.

È proprio attraverso questa proprietà che si legge e si aggiorna il contenuto della variabile. Questo passaggio è importante: ref() non restituisce un valore semplice, ma un contenitore reattivo.

```
const message = ref('hello world!');
```

```
<template>
  <h1>{{ message }}</h1>
```

defineProps

Si utilizza per dichiarare le proprietà che un componente riceve dal suo genitore, consentendo di stabilire quali dati esterni un componente può ricevere e in quale forma essi debbano presentarsi.

Il funzionamento è semplice: si passa a defineProps() un oggetto di configurazione in cui vengono indicate le proprietà ammesse, insieme ai loro tipi e ad eventuali valori predefiniti.

defineEmits

Viene utilizzata per dichiarare e gestire gli eventi personalizzati che un componente figlio può emettere verso il componente genitore.

In altre parole, consente di stabilire in maniera chiara quali segnali il componente è in grado di inviare, garantendo così un flusso di dati ordinato e leggibile.

```
const emit = defineEmits(['remove-task']);
```

Compiler macro

definePropsO e defineEmits() sono cosiddette **compiler macro**:

Ciò significa che sono interpretate e trasformate direttamente dal compilatore di Vue durante la fase di build, e non esistono realmente a runtime.

Questo consente lor di essere estremamente efficienti e di integrarsi in modo naturale con la sintassi reattiva di <script setup>.