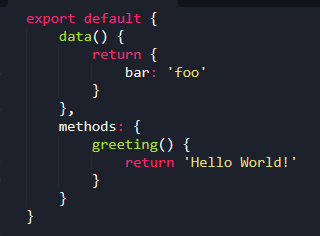
A API Composition é uma sintaxe alternativa para escrever componentes, totalmente opcional.

Essa é uma comparação entre um arquivo escrito com options API:



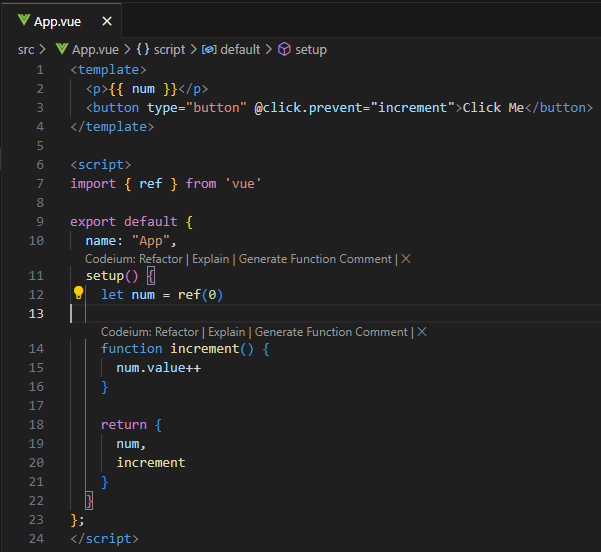
E outro arquivo que faz a mesma coisa, escrito com Composition API:



Uma das razões em se usar o composition API é que ele oferece melhor suporte ao TypeScript. Outra razão é sua melhor organização. Melhor reutilização também.

**MIXINS**

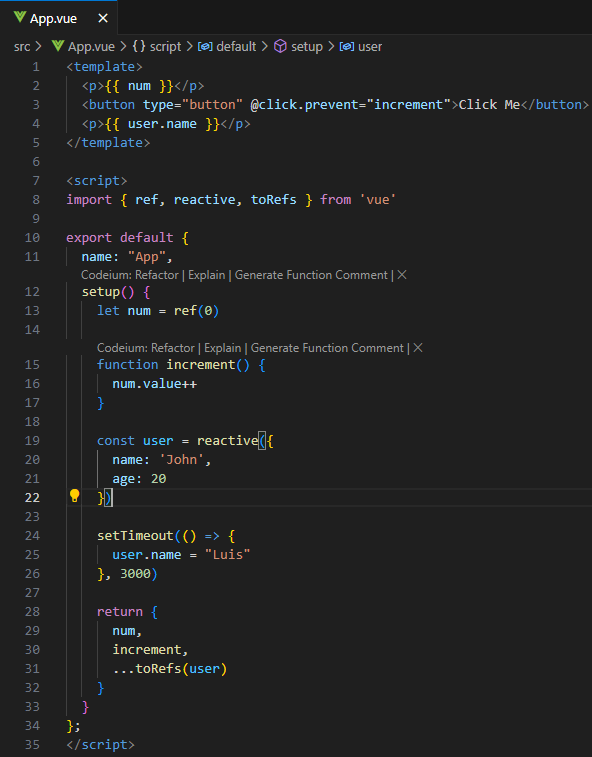
**REFERÊNCIAS REATIVAS**

****

**A FUNÇÃO REATIVA**

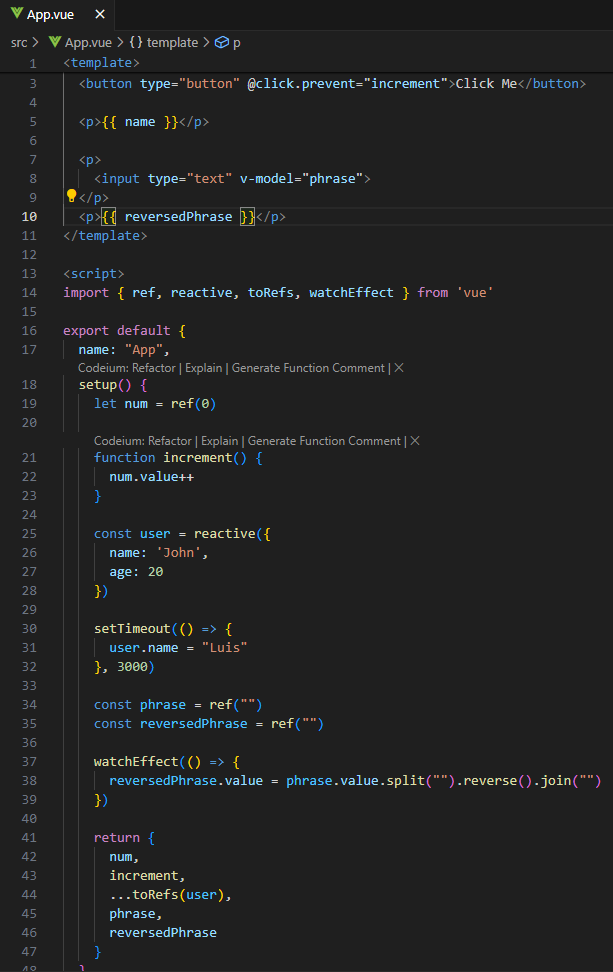
A grande diferença entre a função reativa e a ref() é que a primeira serve para tornar reativos os valores primitivos, como strings, numeros e booleanos. Para objetos, utiliza-se a ref().

A função reativa permite que os valores das propriedades sejam acessados diretamente, no caso das ref() os valores precisam ser acessados via propriedade value.



**WATCHERS AND COMPUTED PROPERTIES**

Uma demonstração de seu funcionamento é mostrar ao contrário um valor digitado em um input, utilizando a watchEffect:



**LIFECYCLE FUNCTIONS**

**onBeforeMount()**

**onMounted()**

**onBeforeUpdate()**

**onUpdated()**

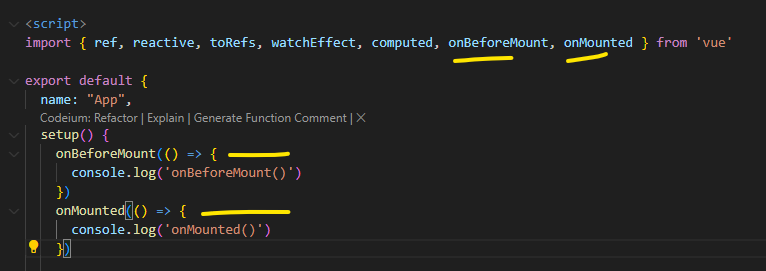
**beforeDestroy()**

**destroyed()**

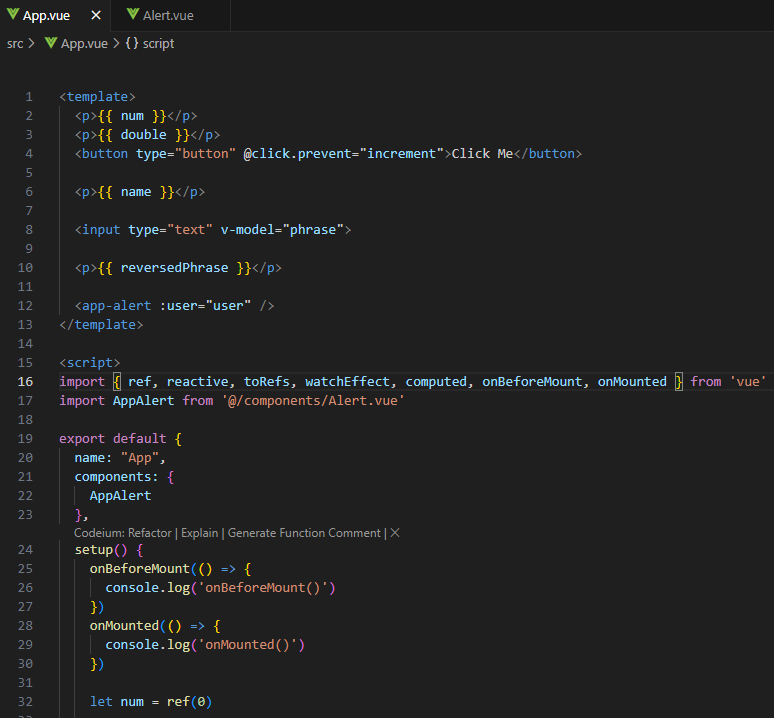
**onActivated()**

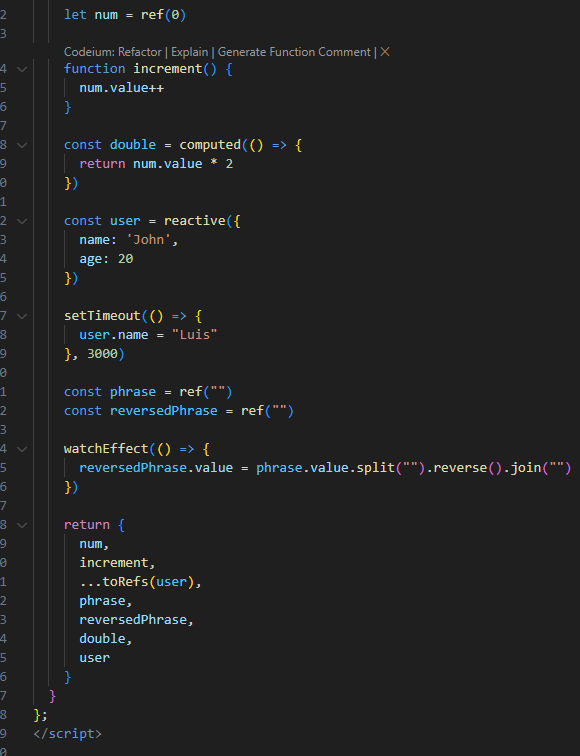
**onDeactivated()**

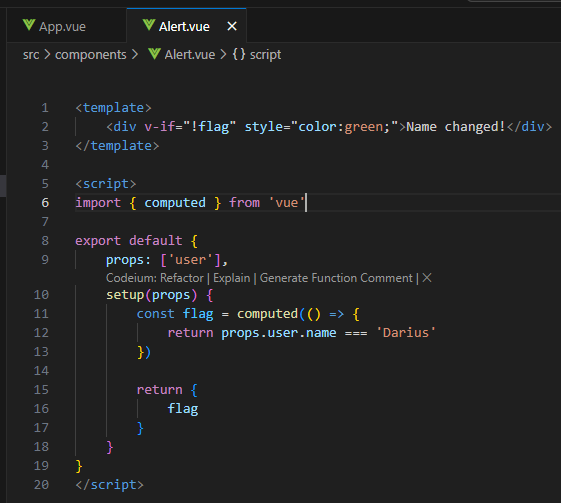
**\*As funções de beforeCreate() e created() não estão incluídas por que a função setup() roda após o beforeCreate, mas antes do created().**

****

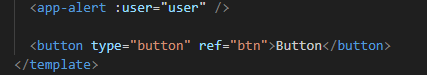
**PROPS**

****

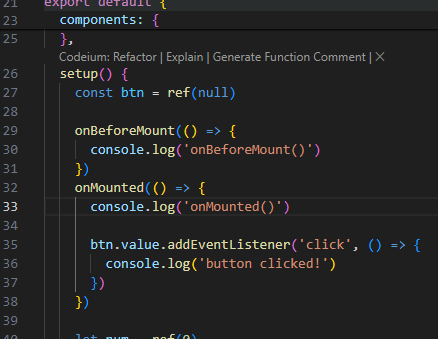
****

****

**TEMPLATE REFS**



Adicionei um botão e atribuí uma ref a ele.



**EMITINDO EVENTOS**

Setup() permite um segundo parâmetro além de props, que é o context, e com ele é possível emitir.

**ROUTER HOOKS**

**PINIA HOOKS**

**VERIFYING REACTIVITY**

**SETUP ATTRIBUTE**