06

Para saber mais: Options API e Composition API

Sabia que, até agora, nós estivemos utilizando um dos dois estilos possíveis de desenvolvimento no Vue?

O estilo de API que estamos usando se chama **Options API** (API de Opções, em português), e nela nós usamos **opções** para definir a **lógica** de um componente, como `data()`, `methods`, `props`, `created()`, entre outras.

Porém, existe um outro estilo de API que também é usado no Vue, que se chama **Composition API** (API de Composição, em português). Esse estilo é um recurso nativo no Vue 3 e no Vue 2.7.

Vamos entender um pouco a diferença entre esses dois estilos? Vamos usar o código atual do

`<script>` do `ConteudoPrincipal` como exemplo:

```
<script lang="ts">
import SelecionarIngredientes from './
import SuaLista from './SuaLista.vue';

export default {
  data() {
    return {
      ingredientes: [] as string[]
    };
  },
  components: { SelecionarIngredientes
  methods: {
    adicionarIngrediente(ingrediente:
      this.ingredientes.push(ingredien
    },
    removerIngrediente(ingrediente: st
      this.ingredientes = this.ingredi
    },
  }
}
```

```
</script>
```

COPIAR CÓDIGO

Para usar a Composition API em um componente, o ponto de entrada é o método chamado `setup()`, disponível como uma das propriedades do objeto do componente (o objeto do `export default`). Enquanto as propriedades que viemos usando até agora fazem parte da Options API, o `setup()` é o único método que faz parte da Composition API.

Vamos conferir o equivalente ao código acima usando a Composition API:

```
<script lang="ts">
import { ref } from 'vue';
import SelecionarIngredientes from './
import SuaLista from './SuaLista.vue';
```

```
export default {
  setup() {
    const ingredientes = ref<string[]>

    function adicionarIngrediente(ingr
      ingredientes.value.push(ingredie
    }
    function removerIngrediente(ingred
      ingredientes.value = ingrediente
    }

    return {
      ingredientes,
      adicionarIngrediente,
      removerIngrediente
    }
  },
  components: { SelecionarIngredientes
}
</script>
```

COPIAR CÓDIGO

Vamos passar pelos seguintes pontos para conferir as diferenças de código:

- Quase todas as opções do objeto foram removidas, com exceção de `components`. No lugar delas, foi adicionado o método `setup()`, o ponto de entrada da Composition API;
- O estado `ingredientes` agora foi definido com o código `const ingredientes = ref<string[]>([])`. Note que a função `ref()` foi importada do pacote `vue`. Esse método **cria uma variável reativa**, da mesma forma que as propriedades retornadas no `data()` também eram reativas. Além disso, note que é possível definir a tipagem do estado usando generics no método;
- As funções antes declaradas nos `methods` agora são funções normais declaradas dentro de `setup()`;
- Para **acessar ou modificar** o valor do estado `ingredientes`, agora é necessário escrever

`ingredientes.value` em vez de `this.ingredientes`;

- Por fim, o estado e as funções são retornados para o `setup()` dentro de um objeto. Isso vai expor essas informações para o template do componente.

> Leia a seção [Why Refs? (https://vuejs.org/guide/essentials/reactivity-fundamentals.html#why-refs)](https://vuejs.org/guide/essentials/reactivity-fundamentals.html#why-refs) para entender por que é necessário escrever `.value` para acessar o valor de variáveis reativas.

E com isso, o código acima já utiliza perfeitamente a Composition API e funciona exatamente igual a antes! Quando você já entende conceitos do Vue como estado, props, eventos, entre outros, a transição de um estilo para o outro é mais suave e facilitada.

Vantagens e desvantagens

> Mas Evaldo, quais as vantagens de se usar a Composition API? Existem desvantagens em relação à Options API?

A Options API é considerada mais fácil de entender e de se desenvolver para quem está iniciando os estudos em Vue ou em frameworks front-end no geral, pois já possui opções reservadas para diferentes funcionalidades, o que torna o código legível e bem documentado.

A Composition API não possui esses blocos de separação, deixando todo o código dentro do `setup()`. É um estilo diferente de codificação, e pode ser preferido ou não, dependendo de quem está desenvolvendo.

No entanto, um fato é que a Composition API dá mais liberdade para uso dos recursos do Vue, aumentando suas possibilidades de uso, e por esse motivo ela é mais recomendada para projetos de maior porte.

Além disso, a Composition API se integra melhor com o TypeScript, além de tornar o código mais sucinto na maioria dos casos, como veremos logo abaixo.

Contudo, é importante frisar que não há planos da equipe do Vue para descontinuar a Options API! No mercado, diferentes empresas usam diferentes estilos. A Options API consegue fazer praticamente tudo que a Composition API faz, salvo casos de uso muito incomuns onde realmente é necessário recorrer à Composition API.

A documentação do Vue ensina todos os seus conceitos das duas formas. Uma vez que você tenha aprendido um conceito em um dos estilos, basta revisitar a página da documentação referente a esse conceito e alternar o estilo de API para aprendê-lo com outra sintaxe.

Acesse a seção API Styles (https://vuejs.org/guide/introduction.html#api-styles) e a página Composition API FAQ (https://vuejs.org/guide/extras/composition-api-faq.html) da documentação para conferir em detalhes as diferenças entres os dois estilos.

Além disso, da mesma forma que a documentação possui uma página ensinando a usar TypeScript com Options API (https://vuejs.org/guide/typescript/options-api.html), ela também possui uma que ensina a usar TypeScript com Composition API (https://vuejs.org/guide/typescript/composition-api.html).

Usando `<script setup>`

Talvez você não tenha notado uma diferença significativa ao migrar o código da Options API para a Composition API. Na verdade, ele ficou até com algumas linhas extras!

Justamente para evitar muito código repetido, a Composition API é comumente utilizada com um recurso que a deixa mais sucinta e melhora a experiência de desenvolvimento. Esse recurso é o atributo `setup` que pode ser adicionado no `<script>`.

Com isso, algumas mudanças podem ser feitas no código. Confira ele reescrito:

```
<script setup lang="ts">
import { ref } from 'vue';
import SelecionarIngredientes from './
import SuaLista from './SuaLista.vue';

const ingredientes = ref<string[]>([])
```

```
function adicionarIngrediente(ingredie
  ingredientes.value.push(ingrediente)
}
function removerIngrediente(ingredient
  ingredientes.value = ingredientes.va
}
</script>
```

COPIAR CÓDIGO

E olha só! Como mágica, o código ficou bem menor e sucinto. Vamos conferir as mudanças?

- Com a adição do atributo `setup`, não é mais necessário realizar o `export default` e nem escrever o método `setup()`. É como se agora todo o código do `<script>` já estivesse dentro do `setup()`, por baixo dos panos.
- A grande vantagem do `<script setup>` é não precisar mais retornar um objeto com as informações que queremos expor ao template do componente. Ao invés disso,

todas as variáveis declaradas e importadas estão automaticamente disponíveis para o template, como o estado `ingredientes`, as funções e até mesmo os componentes!

- Note que, como os componentes agora estão disponíveis para o template, também não precisamos mais da opção `components` que estávamos usando até agora.

O Vue recomenda fortemente utilizar o `<script setup>` ao se desenvolver com a Composition API, por tornar o código mais simples e direto. Por esse motivo, as páginas da documentação em Composition API ensinam os conceitos usando principalmente o `<script setup>`, mas também ensinam sem o `setup` quando necessário.