Vue3 升级了哪些重要功能

Vue 3 升级了许多重要功能，以下是一些主要的改进和优化：

1. 响应式系统： Vue 3 使用了 Proxy 对象替代 Vue 2 中的 Object.defineProperty，使得响应式系统更加高效和灵活。这个新的响应式系统可以追踪更细粒度的依赖关系，提供了更好的性能和更细致的响应式控制。
2. Composition API： Vue 3 引入了 Composition API，这是一种新的组合式 API 风格，可以更好地组织和复用组件逻辑。通过 Composition API，开发者可以根据功能而不是组件层次结构来组织代码，使代码更加清晰和可维护。
3. TypeScript 支持： Vue 3 对 TypeScript 的支持更加完善。Vue 3 的代码库已经使用 TypeScript 重写，并且提供了更好的类型推断和类型检查。
4. Tree-shaking 支持： Vue 3 改进了 Tree-shaking 的支持，可以更好地优化打包大小，只包含使用到的代码。
5. 性能优化： Vue 3 在性能方面进行了一些改进，包括更快的渲染速度和更小的包大小。Vue 3 使用了新的响应式系统和编译器优化，提高了渲染性能。
6. 逻辑复用： Vue 3 的 Composition API 提供了更好的逻辑复用能力，可以更方便地将逻辑提取为可复用的函数。
7. 开发体验： Vue 3 提供了更好的开发体验，包括更好的错误提示、更好的调试工具和更好的开发工具支持。

总的来说，Vue 3 在性能、可维护性、TypeScript 支持和开发体验等方面都有显著的改进和优化。

具体的升级如下：

- createApp
- emits 属性
- 生命周期
- 多事件
- Fragment
- 移除 .sync
- 异步组件的写法
- 移除 filter
- Teleport
- Suspense
- Composition API

（1）createApp

```
// vue 2.x
const app = new Vue({ /* 选项 */ });
```

```
Vue.use( /* ... */);
Vue.mixin( /* ... */);
Vue.component( /* ... */);
Vue.directive( /* ... */);

// vue 3.x
const app = Vue.createApp({ /* 选项 */ });

app.use( /* ... */);
app.mixin( /* ... */);
app.component( /* ... */);
app.directive( /* ... */);
```

（2）emits 属性

Vue3 中，子组件中需要先申明 `emits`。

（3） 多事件处理

```
<!-- 定义 one 和 two 两个函数 -->
<button @click="one($event), two($event)">submit</button>
```

（4）Fragment

在 Vue2 中单个根元素规则，`template` 里每个组件必须只有一个根元素。

Vue3 有了片段的支持，因此组件不再需要根节点。

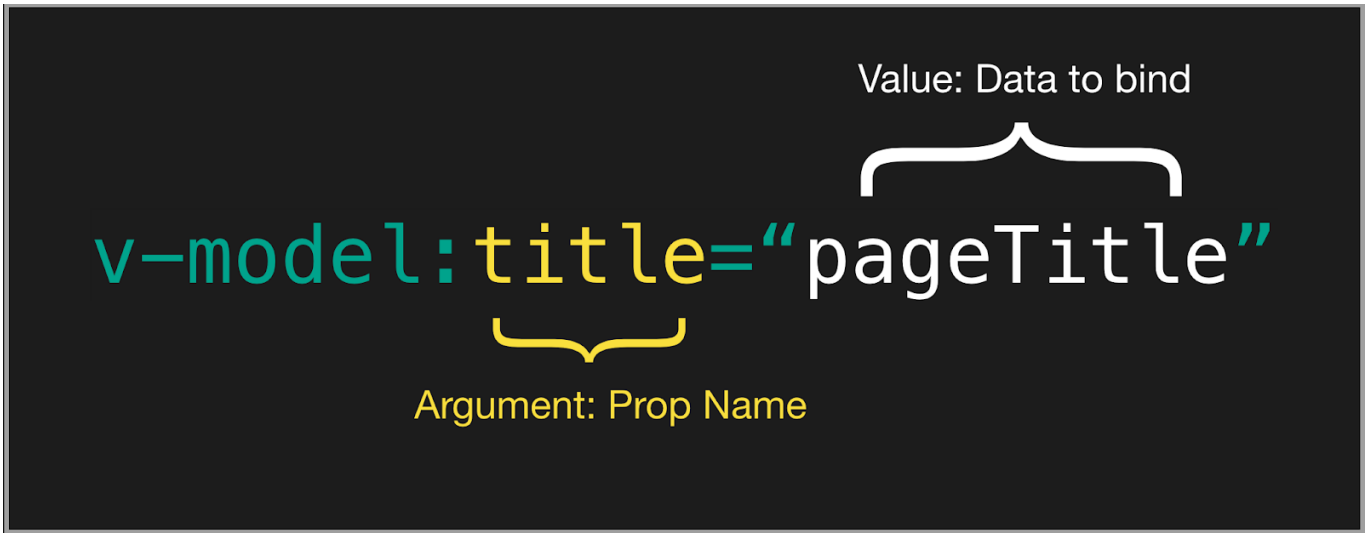
```
<!-- vue2 组件模版 -->
<template>
  <div class="blog-post">
    <h3>{{ title }}</h3>
    <div v-html="content"></div>
  </div>
</template>

<!-- vue3 组件模版 -->
<template>
  <h3>{{ title }}</h3>
  <div v-html="content"></div>
</template>
```

（5）移除 .sync

```
<!-- vue2 -->
<ChildComponent :title="pageTitle" @update:title="pageTitle = $event" />
<!-- 简写 -->
<ChildComponent :title.sync="pageTitle" />
```

```
<!-- vue3 -->
<ChildComponent :title="pageTitle" @update:title="pageTitle = $event" />
<!-- 简写 -->
<ChildComponent v-model:title="pageTitle" />
```



(6) 异步组件的写法

- 新的 `defineAsyncComponent` 助手方法，用于显式地定义异步组件
- `component` 选项被重命名为 `loader`
- Loader 函数本身不再接收 `resolve` 和 `reject` 参数，且必须返回一个 Promise

```
// vue2
const asyncModal = {
  component: () => import('./Modal.vue'),
  delay: 200,
  timeout: 3000,
  error: ErrorComponent,
  loading: LoadingComponent
}


// vue3
import { defineAsyncComponent } from 'vue'
import ErrorComponent from './components/ErrorComponent.vue'
import LoadingComponent from './components/LoadingComponent.vue'

// 带选项的异步组件
const asyncModalWithOptions = defineAsyncComponent({
  loader: () => import('./Modal.vue'),
  delay: 200,
  timeout: 3000,
  errorComponent: ErrorComponent,
  loadingComponent: LoadingComponent
})
```

```
// 2.x 版本
const oldAsyncComponent = (resolve, reject) => {
  /* ... */
}

// 3.x 版本
const asyncComponent = defineAsyncComponent(
  () =>
    new Promise((resolve, reject) => {
      /* ... */
    })
)
```

（7）移除 filter

从 Vue 3.0 开始，过滤器已移除，且不再支持。建议用方法调用或计算属性来替换它们。

（8）Teleport

`<Teleport>` 是一个内置组件，它可以将一个组件内部的一部分模板“传送”到该组件的 DOM 结构外层的位置去。

```
<button @click="open = true">Open Modal</button>

<Teleport to="body">
  <div v-if="open" class="modal">
    <p>Hello from the modal!</p>
    <button @click="open = false">Close</button>
  </div>
</Teleport>
```

（9）Suspense

```
<Suspense>
  <!-- 具有深层异步依赖的组件 -->
  <Dashboard />

  <!-- 在 #fallback 插槽中显示 “正在加载中” -->
  <template #fallback>
    Loading...
  </template>
</Suspense>
```

（10）Composition API

- reactive
- ref

- readonly
- watch 和 watchEffect
- setup
- 生命周期钩子函数