



马上就好



# Vue Macros

介绍及基本原理

三咲智子




2022-12-10


# 三咲智子 Kevin Deng


Vue / VueUse 团队成员，Element Plus 主要贡献者  
为 Vue / Vite 及周边生态做贡献


大三学生，目前正在为全职开源努力



 [sxzz](#)

 [sanxiaozhizi](#)

 [三咲智子](#)

 [三咲智子](#)



- 什么是宏 macro ?
- Vue Macros 是什么?
- Vue Macros 功能介绍
- 对 Vue Macros 未来的计划

# `<script setup>`

Vue 3.2 引入了 `<script setup>` 语法

```
<script>
import { ref } from 'vue'

export default defineComponent({
  setup() {
    const msg = ref('')

    return {
      msg,
    }
  }
})
</script>

<template>
  <span>{{ msg }}</span>
</template>
```



```
<script setup>
import { ref } from 'vue'

const msg = ref('')
</script>

<template>
  <span>{{ msg }}</span>
</template>
```

# 如何定义组件的属性？

使用 `\defineProps\`, `\defineEmits\`, `\defineExpose\` 等宏

```
<script setup>
// 这里是 setup 代码
defineProps(['title', 'count'])
const emit = defineEmits(['change'])

emit('change', 'new value')
</script>
```

```
const __sfc__ = {
  props: ["title", "count"],
  emits: ["change"],

  setup(__props, { emit }) {
    // 这里是 setup 代码
    emit("change", "new value");

    return {};
  },
};
export default __sfc__;
```

# 使用 TypeScript 定义

TypeScript 类型声明来定义 props 和 emits

```
<script setup lang="ts">
defineProps<{
  foo?: string
}>()
const emit = defineEmits<{
  (evt: 'change', value: string): void
}>()

emit('change', 'new value')
</script>
```

```
const __sfc__ = {
  props: {
    foo: { type: String, required: false },
  },
  emits: ["change"],
  setup(__props, { emit }) {
    emit("change", "new value");

    return {};
  },
};

export default __sfc__;
```







什么是 **Vue Macros** ?



# 什么是 Vue Macros?



Vue Macros 实现了一系列还没有被 Vue 官方实现的提案 RFC 或来自社区主意。

- ✨ 为 Vue 扩展更多的宏和语法糖；
- ❤️ 开箱即用地支持 Vue 2.7 和 Vue 3；
- 🛡️ 类型安全
  - 支持  TypeScript 和  Volar 插件；
- ⚡ 支持多种打包器
  - 同时支持  Vite 3/4,  Webpack,  Vue CLI,  Rollup 2/3,  esbuild 等。使用  `unplugin` 驱动。

# defineOptions

RFC [vuejs/rfcs #430](#)

在一个 SFC 中，避免使用两个 `<script>` 标签

```
<script>
// index.vue
import { defineComponent, ref } from 'vue'

export default defineComponent({
  name: 'Counter',
  inheritAttrs: false,
})
</script>

<script setup>
const count = ref(0)
</script>
```



```
<script setup>
// index.vue
import { ref } from 'vue'

defineOptions({
  name: 'Counter',
  inheritAttrs: false,
})

const count = ref(0)
</script>
```

# hoistStatic

PR [vuejs/core #5752](#)

---

提升静态常量到顶级作用

```
<script setup>
// index.vue

const componentName = 'Counter'
defineOptions({
  name: componentName,
})

if (someError) {
  console.error(`${componentName} error!`)
}
</script>
```

```
<script>
// index.vue
import { defineComponent, ref } from 'vue'

const componentName = /* hoist-static */ fn()

export default defineComponent({
  name: componentName,
  setup() {
    if (someError) {
      console.error(`${componentName} error!`)
    }
  }
})
</script>
```

# defineModel

简化重复的 props 和 emits 声明，由  VueUse 驱动。

```
<script setup lang="ts">
const props = defineProps<{
  modelValue: string
  count: number
}>()

const emit = defineEmits<{
  (evt: 'update:modelValue', value: string): void
  (evt: 'update:count', value: number): void
}>()

emit('update:modelValue', 'newValue')
emit('update:count', props.count + 1)
</script>
```



```
<script setup lang="ts">
const { modelValue, count } = defineModel<{
  modelValue: string
  count: number
}>()

modelValue.value = 'newValue'
count.value++
</script>
```

# Vue 2 中使用 defineModel

统一模式下会自动把 prop 和 event 的名称转换成 Vue 2 的规则。

```
prop: `modelValue` → `value`  
event: `update:modelValue` → `input`
```

✨ 同样的代码，都可以运行在 Vue 2 和 3 中！

# defineModel 与 Reactivity Transform

开启 Reactivity Transform 后，代码还可以更短一些！不由  VueUse 驱动。

```
<script setup lang="ts">
const { modelValue, count } = defineModel<{
  modelValue: string
  count: number
}>()
```

```
modelValue.value = 'newValue'
count.value++
</script>
```



```
<script setup lang="ts">
let { modelValue, count } = $defineModel<{
  modelValue: string
  count: number
}>()
```

```
modelValue = 'newValue'
count++
</script>
```

# setupComponent

在 JS/TS 文件中使用宏

```
export const App = defineSetupComponent(() => {  
  defineProps<{  
    foo: string  
 }>()  
  
  defineEmits<{  
    (evt: 'change'): void  
 }>()  
  
  defineOptions({  
    name: 'App',  
  })  
  
  // ...  
})
```

```
export const App: SetupFC = () => {  
  defineProps<{  
    foo: string  
 }>()  
  
  defineEmits<{  
    (evt: 'change'): void  
 }>()  
  
  defineOptions({  
    name: 'App',  
  })  
}
```

# setupSFC

在 JS/TS 文件中使用宏

```
// Foo.setup.tsx
defineProps<{
  foo: string
}>()

defineEmits<{
  (evt: 'change'): void
}>()

export default () => (
  <div>
    <h1>Hello World</h1>
  </div>
)
```



# namedTemplate

vuejs/core #6898

---

定义子模板，可在主模板中引用子模板的内容

```
<script setup>
const direction = 'top'
</script>

<template name="tab-header">
  <span>Tab header</span> <!-- ..... -->
</template>

<template name="tab-body">
  <span>Tab body</span> <!-- ..... -->
</template>

<template>
  <template v-if="direction === 'top'">
    <template is="tab-header" /> <template is="tab-body" />
  </template>
  <template v-else>
    <template is="tab-body" /> <template is="tab-header" />
  </template>
</template>
```

# 一些小的语法糖 偷懶!

## defineRender

```
<script setup lang="tsx">
// 直接传递 JSX
defineRender(
  <div>
    <span>Hello</span>
  </div>
)
```

```
// 或者渲染函数
defineRender(() => {
  return (
    <div>
      <h1>Hello World</h1>
    </div>
  )
})
</script>
```

## shortVmodel

RFC vuejs/rfcs #395

```
<template>
  <input $="msg" />
  <input ::="msg" />
  <!-- => <input v-model="msg" /> -->
  <demo $msg="msg" />
  <!-- => <input v-model:msg="msg" /> -->
</template>
```

## shortEmits

```
<script setup lang="ts">
const emits = defineEmits<{
  (evt: 'update:modelValue', value: string): void
  (evt: 'update', value: string): void
}>()
```

// `ShortEmits` 或简写为 `SE`, 可以使用元组或方法定义。

```
const emits = defineEmits<SE<{
  'update:modelValue': [val: string]
  update(val: string): void
}>>()
</script>
```

# betterDefine

issue [vuejs/core #4294](#)

支持在 `<script setup>` 导入 TS 类型来定义 props 和 emits  
Vue 官方的支持会在 3.3 发布

交集 Intersections

```
<script setup lang="ts">
import { FooProps } from './types'

defineProps<FooProps & {
  title?: string
}>()
</script>
```

继承 Extends

```
<script setup lang="ts">
import { FooProps } from './types'

interface Props extends FooProps {
  title?: string
}

defineProps<Props>()
</script>
```

`defineEmits`

```
<script setup lang="ts">
import { FooEmits } from './types'

interface Emits extends FooEmits {
  (evt: 'change', val: string): void
}

defineEmits<Emits>()
</script>
```

# 计划 `@vue-macros/api`

公开的核心 API

- 分析 SFC 组件的基本信息
- 对 props、emits 等进行增删改查
- 目前只支持用宏 + TS 类型声明

 WIP

- 运行时定义 / Runtime
- `expose / name / inheritAttrs`
- 代码转换
- ...

# 未来的 defineMacro

issue #6392

- 不再局限于 Vue Macros 提供的宏；
- 可以使用社区发布的宏；
- 通过 `@vue-macros/api` 提供核心能力；
- 更灵活地定义组件的属性 (props, emits...)

`unplugin-vue-router` 的 `definePage`

```
<script setup>
definePage({
  name: 'my-own-name',
  path: '/absolute-with-:param',
  alias: ['/a/:param'],
})
</script>
```

# 谁在使用 *Macros* VUE ?



Element Plus

``defineOptions` / `hoistStatic``



Elk 鹿鸣

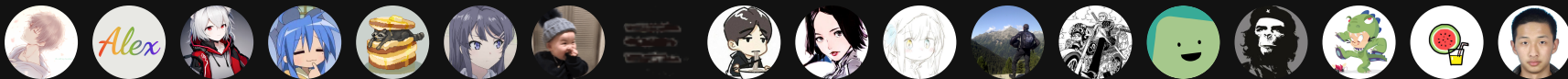
``defineOptions` / `defineModel` / `defineSlots``



Nuxt 3 一行配置，开箱即用

# 贡献者

💕 感谢所有贡献者的参与!



♥ Silver Sponsors



patak



Evan You

♥ Bronze Sponsors



Lo



Luke Diebold

♥ Sponsors



KallyDev



C.Y.Kun



Shareworks



Magren



Alex



mrcat



cyn



John

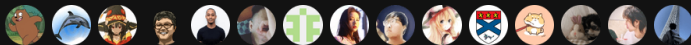


AingCyan

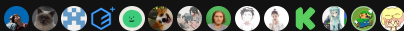


Daniel Roe

♥ Backers




Past Sponsors



♥ 在 GitHub 赞助我!



# 感谢

幻灯片可稍后在  [github.com/sxzz/talks](https://github.com/sxzz/talks) 浏览

💖 感谢 [Anthony Fu](#), 幻灯片由  Slidev 强力驱动!