

Introduction to

Vue Macros

Explore and extend more macros and syntax sugar to Vue.

KEVIN DENG (三咲智子)



Kevin Deng (三咪智子)

Team member of Vue, VueUse and Element Plus.

College student currently working towards a full time job on open source.



💆 sanxiaozhizi

S sxzz.moe (Chinese)



What's **Vue Macros**?

What is Vue Macros?

It is a series of proposals or ideas that have not been officially implemented by Vue.

- **†** More syntax sugar and macros
 - Explore and extend more macros and syntax sugar to Vue.
- Out-of-the-box and good compatibility
 - Supports both Vue 2.7+ and Vue 3 out-of-the-box.
- Type Safe
 - Full Ts TypeScript and V Volar support.
- Multiple bundlers are supported
 - Supports Vite, Webpack, Vue CLI, Rollup, esbuild and more, powered by unplugin.

Talk is cheap...

Show me the code!

defineOptions

Avoiding two `<script>` tags in the one SFC. RFC

```
<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>
```

defineModel

Simplify verbose declaration of props and emits. Thanks to U VueUse.

```
<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>
```

defineModel on Vue 2

With unified mode, it will transform prop and event names for Vue 2.

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

₩ Write once, run on both Vue 2 and 3!

defineModel with Reactivity Transform

With Reactivity Transform, it can even be simplified further.

```
<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>
```

betterDefine

Support imported types in `<script setup>` type-based macros. Official support is expected in Vue 3.3.

Intersections

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

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

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>
```

Let's have a try!

Contributors

Thank you to all the contributors!

























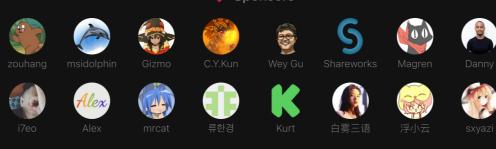




♥ Silver Sponsors



Sponsors









♥ Sponsor me at GitHub

Thank You!

Slides can be found on 🖸 github.com/sxzz/talks