Patterns of VueUse

Patterns and best practices we have learnt during the past two years of building VueUse.

ANTHONY FU

Vue Fes Japan Online 2022
Oct. 16th 2022

Anthony Fu

Core team member of Vue, Vite and Nuxt.

Creator of VueUse, Vitest, UnoCSS, Slidev and Type Challenges.

Working at NuxtLabs.

- antfu
- antfu7
- antfu.me

Special Sponsor

NuxtLabs

Platinum Sponsors

Evan You
Astro
Akamai Community

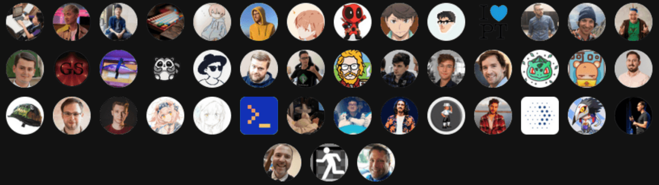
Gold Sponsors

Leniolabs_
Vue Mastery
Vercel
Matej yangwao

Silver Sponsors

Ben Hong
Jess
Antony
Stanislas
Matt
Mark Bokil

Sponsors



Backers

Sponsor me at GitHub

What’s VueUse?

Collection of Vue Composition Utilities

v9.1.1 1.1M/month docs & demos 248 functions

Stars 12k

Works for both Vue 2 and 3

Tree-shakeable ESM

CDN compatible

TypeScript

Rich ecosystems

State

createGlobalState
createInjectionState
createSharedComposable
useAsyncState
useDebouncedRefHistory
useLastChanged
useLocalStorage
useManualRefHistory
useRefHistory
useSessionStorage
useStorage
useStorageAsync
useThrottledRefHistory

Elements

useActiveElement
useDocumentVisibility
useDraggable
useDropZone
useElementBounding
useElementSize
useElementVisibility
useIntersectionObserver
useMouseInElement
useMutationObserver
useResizeObserver
useWindowFocus

Functions

Core	State · Elements · Browser · Sensors · Network · Animation · Component · Watch · Reactivity · Array · Time · Utilities
Add-ons	Electron · Firebase · Head · Integrations · Math · Motion · Router · RxJS · SchemaOrg · Sound
Sort by	Category · Name · Updated
Filters	Has Component · Has Directive

Search...

State

- createGlobalState - keep states in the global scope to be reusable across Vue instances
- createInjectionState - create global state that can be injected into components
- createSharedComposable - make a composable function usable with multiple Vue instances
- useAsyncState - reactive async state
- useDebouncedRefHistory - shorthand for `useRefHistory` with debounced filter
- useLastChanged - records the timestamp of the last change
- useLocalStorage - reactive LocalStorage
- useManualRefHistory - manually track the change history of a ref when the using calls `commit()`
- useRefHistory - track the change history of a ref
- useSessionStorage - reactive SessionStorage
- useStorage - reactive LocalStorage/SessionStorage
- useStorageAsync - reactive Storage in with async support

State of VueUse

Until September 9th, 2022

1.1M	Monthly downloads on NPM
435K	Monthly pageviews on docs
36.8K	Open Source projects dependents
11.6K	Stars on GitHub
1000	Days since the first commit
318	Contributors to the core package
248	Composable functions
13	Team members
10	Addons packages

What We Have Learnt?

Constructing Connections

Constructing Connections

Vue and the Composition API

- State drives UI - single source of truth
- Changes on state are auto reflected - reactivity
- In `` `template` ``, we build connectons between state and UI
- In `` `setup()` `` function, we build connections between data and logics

Passing Refs as Arguments

Writing a composable function

	Implementation	Usage

Plain function	`function add(a: number, b: number) {`
	`  return a + b`
	`}`
		`let a = 1`
		`let b = 2`
		
		`let c = add(a, b) // 3`

Accpets refs, returns a reactive result.	`function add(a: Ref<number>, b: Ref<number>) {`
	`  return computed(() => a.value + b.value)`
	`}`
		`const a = ref(1)`
		`const b = ref(2)`
		
		`const c = add(a, b)`
		`c.value // 3`

Accpets both refs and plain values.	`function add(`
	`  a: Ref<number> | number,`
	`  b: Ref<number> | number`
	`) {`
	`  return computed(() => unref(a) + unref(b))`
	`}`
		`const a = ref(1)`
		
		`const c = add(a, 5)`
		`c.value // 6`

Implementation of `ref`

Dive into Vue's codebase

```
function ref(input) {
  return isRef(input)
    ? input
    : createRef(input)
}
```

Which means:

```
const foo = ref(123)
const bar = ref(foo)

foo === bar // true
```

Tips

`ref()` forwards existing ref

Implementation of `unref`

Dive into Vue’s codebase

```js
function unref(input) {
  return isRef(input)
    ? input.value
    : input
}
```

Which means:

```js
const foo = unref(123)

unref === 123 // true
```

Tips

`unref()` forwards plain value

MaybeRef

A custom type helper

```ts
type MaybeRef<T> = Ref<T> | T
```

- When using it as a value, we wrap it with `` `unref()` ``
- When using it as a ref, we wrap it with `` `ref()` ``

For example:

Plain Function

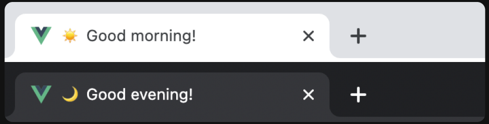
```ts
export function getTimeAgo(time: Date | number | string) {
  return format(time)
}
```

Reactive Function

```ts
export function useTimeAgo(time: MaybeRef<Date | number | string>) {
  return computed(() => format(unref(time)))
}
```

Example

Update page title for light/dark mode



Normal usage

```js
import { useDark, useTitle } from '@vueuse/core'
```

```js
const isDark = useDark()
const title = useTitle()

watch(isDark, () => {
  title.value = isDark.value ? '🌙 Good evening!' : '☀ Good morning!'
})
```

Connection usage

```js
const isDark = useDark()
const title = computed(() => isDark.value ? '🌙 Good evening!' : '☀ Good morning!')

useTitle(title)
```

Check in VueUse: `useTitle`

Taking it Further

Making it more flexible

- `` `computed()` `` converts a function to a ref
- We accpets refs as arguments

In VueUse 9.0, we introduce a new convention:

```
const isDark = useDark()
const title = computed(() => isDark.value ? '🌙 Good evening!' : '☀ Good morning!')

useTitle(title)
```

Turn into:

```
const isDark = useDark()

useTitle(() => isDark.value ? '🌙 Good evening!' : '☀ Good morning!')
```

We call it **"Reactive Getter"**

Reactivity Transform

Learn more at https://vuejs.org/guide/extras/reactivity-transform

```js
let count = $ref(0) // count is a plain value
count = 1
console.log(count)
```

→

```js
let count = ref(0)
count.value = 1
console.log(count.value)
```

Limitation

```js
watch(count, () => { }) // !! this will lose the reactivity
```

```js
watch(() => count, () => { }) // should use a getter function
```

In VueUse with Reactive Getter

```js
useTitle(() => `Count: ${count}`)
```

Reactify

Build connections magically!

VueUse provides an utility function `reactify()` to convert a plain function to reactive one!

```ts
import { reactify } from '@vueuse/core'

function getTimeAgo(time: Date | number | string) {
  return format(time)
}

const useTimeAgo = reactify(getTimeAgo)
```

`unref()` arguments passing to the function and wrap the return with `computed()`

```ts
const date = ref(new Date())

const timeago1 = useTimeAgo(date) // Computed<string>

const timeago2 = useTimeAgo(() => Date.now() - 1000) // Computed<string>
```

Check in VueUse: `reactify`

Side-effect Clean Up

Auto Clean Up for `watch`

And others like `watchEffect` `computed`

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

const count = ref(0)

watch(count, () => {
  console.log('Count: ' + count.value)
})
</script>
```

💡 Tips

When the component get destroyed, `watch()` will be automatically removed.

Clean Up for Custom Composables

```js
export function useEventListener(name, handler) {
  window.addEventListener(name, handler)

  onUnmounted(() => {
    window.removeEventListener(name, handler)
  })
}
```

Tips

Use `onUnmounted()` hook to register side-effect clean up.

Manual Clean Up

`` `watch()` `` will return a stop handler for manual clean up.

```
const count = ref(0)
const stop = watch(count, () => {
  console.log('Count: ' + count.value)
})

count.value += 1 // Count: 1
stop()
count.value += 1 // nothing
```

Manual Clean Up

For Custom Composables

```
export function useEventListener(name, handler) {
  window.addEventListener(name, handler)

  const cleanup = () => {
    window.removeEventListener(name, handler)
  }

  onUnmounted(cleanup)

  return cleanup
}
```

Usage would be:

```
const stop = useEventListener('mousedown', () => {})

stop() // unregister events
```

But…

It could be cumbersome…

For example:

```
function useMouse() {
  const stop1 = useEventListener('mousedown', () => {})
  const stop2 = useEventListener('mouseup', () => {})
  const stop3 = useEventListener('mousemove', () => {})

  const cleanup = () => {
    stop1()
    stop2()
    stop3()
  }

  return cleanup
}
```

Effect Scope

Introduced in Vue 3.1

```js
import { effectScope } from 'vue'

const scope = effectScope()
scope.run(() => {
  const count = ref(0)
  const doubled = computed(() => counter.value * 2)

  watch(doubled, () => console.log(doubled.value))

  useEventListener('mousedown', () => {})
  useEventListener('mouseup', () => {})
  useEventListener('mousemove', () => {})
})

// to dispose all effects in the scope
scope.stop()
```

Learn more at https://vuejs.org/api/reactivity-advanced.html#effectscope

onScopeDispose

For composable to work best with `effectScope`, replace `onUnmounted` with `onScopeDispose`:

```
export function useEventListener(name, handler) {
  window.addEventListener(name, handler)

- onUnmounted(() => {
+ onScopeDispose(() => {
    window.removeEventListener(name, handler)
  })
}
```

This allows the the clean up to be called on scope dispose.

Tips

- Components are special Scopes
- `onUnmounted` is a special `onScopeDispose`

The VueUse Familly

@vueuse/core
Core functions of VueUse

@vueuse/shared
Internal functions of VueUse

@vueuse/components
Renderless components

@vueuse/math
Reactive Math Utilities

@vueuse/head
Document head manager
by @egoist

@vueuse/sound
Composable for playing sound
by @Tahul

@vueuse/motion
Vue components in motion
by @Tahul

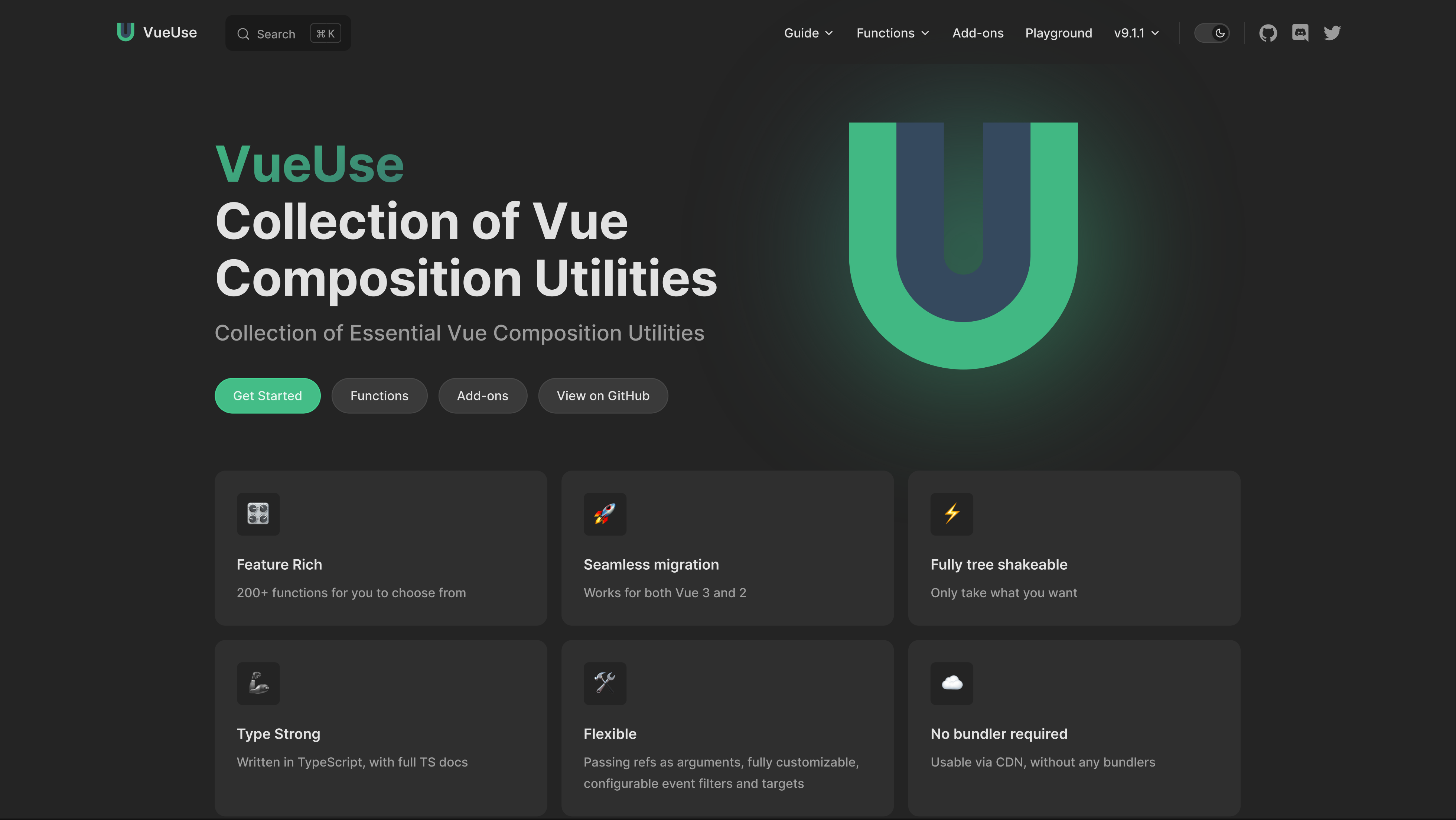
@vueuse/guesture
Composables for interactive
by @Tahul

@vueuse/schema-org
Schema.org graphs for Vue
by @harlan-zw

@vueuse/integrations
Integrations for popular packages

@vueuse/firebase
Firebase integrations

@vueuse/rxjs
RxJS integrations



VueUse Search ⌘K Guide Functions Add-ons Playground v9.1.1

VueUse
Collection of Vue Composition Utilities

Collection of Essential Vue Composition Utilities

Get Started Functions Add-ons View on GitHub

Feature Rich
200+ functions for you to choose from

Seamless migration
Works for both Vue 3 and 2

Fully tree shakeable
Only take what you want

Type Strong
Written in TypeScript, with full TS docs

Flexible
Passing refs as arguments, fully customizable, configurable event filters and targets

No bundler required
Usable via CDN, without any bundlers

Learn more at vueuse.org 💚

Thank You!

Slides can be found on antfu.me