`import { reactive } from 'vue'`

Reactivity System & Composition API of Vue 3 and their applications

Anthony Fu / @antfu / 2020

Anthony Fu

Core Team	Vue.js, Vite.js
Focus	@vue/composition-api, vite
GitHub	/antfu
Twitter	@antfu7
Blog	//antfu.me

//antfu.me/talks/2020-09-26/en

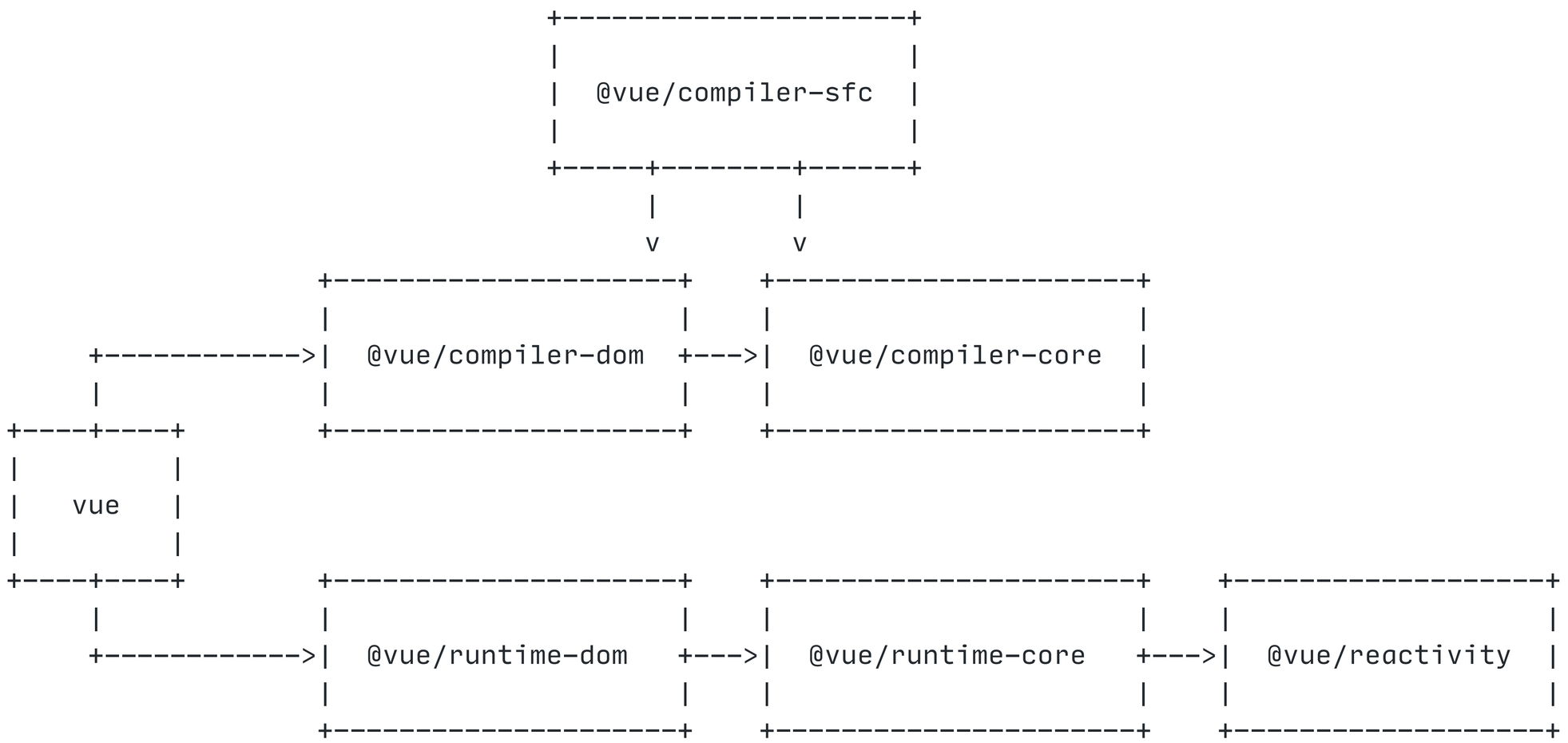
Overview

- Reactivity System & Composition API
- Case Study for Composition API
- Vue 2 & 3 Isomorphic Libraries
- Reactivity System Ecosystem

3.0 Released 🎉

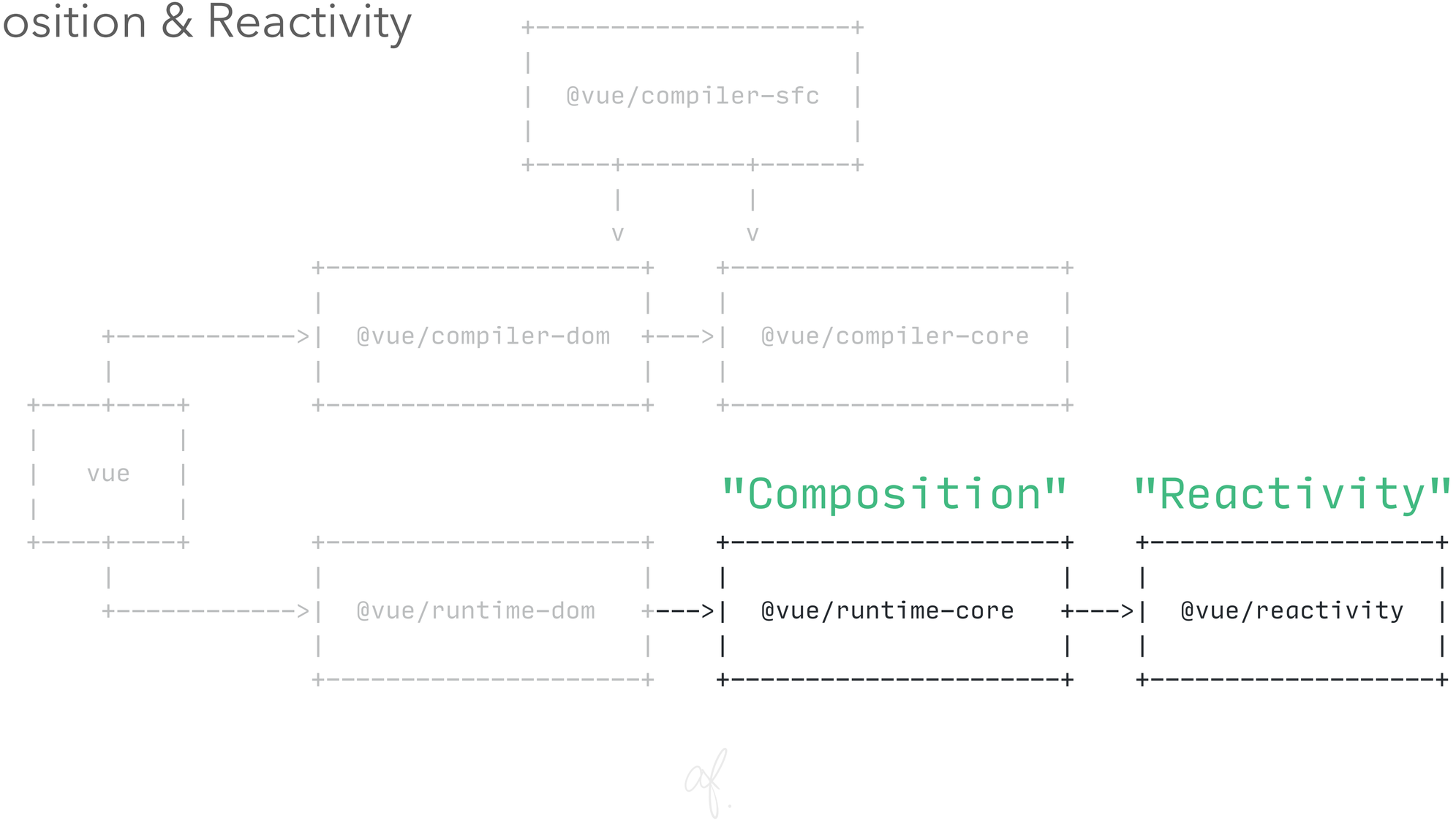
One Piece

Vue 3.0


@vue/compiler-sfc
@vue/compiler-dom
@vue/compiler-core
vue
@vue/runtime-dom
@vue/runtime-core
@vue/reactivity

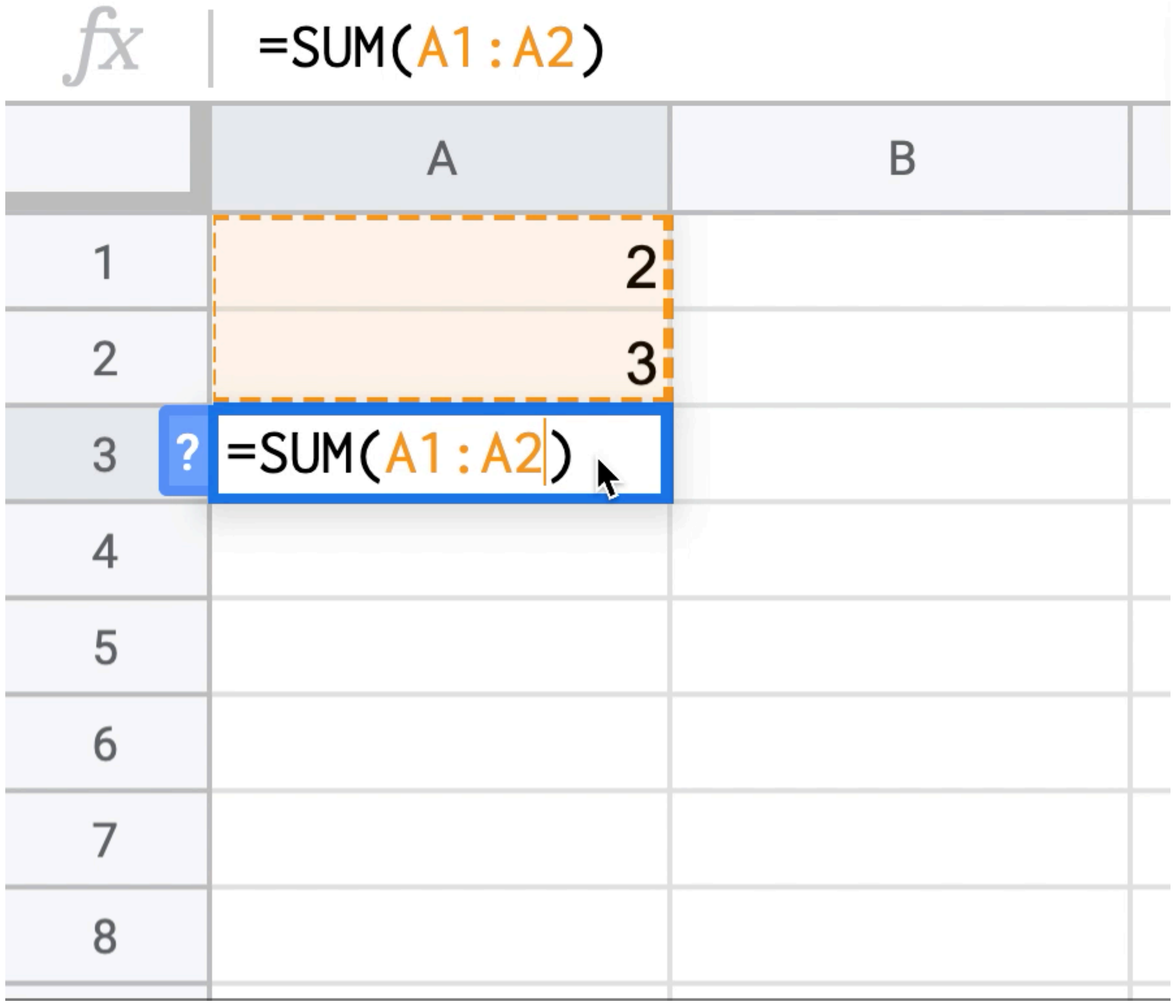
Vue 3.0

Composition & Reactivity



Reactivity

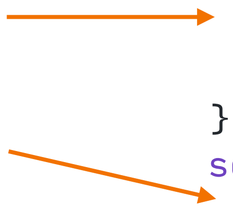
- Dependencies auto collect & update
- New API in Vue 3
  - `ref`
  - `reactive`
  - `effect`
  - `computed`



Reactive

Reactivity

- Implemented with Proxy
- track, trigger



```
const reactive = (target) => new Proxy(target, {
  get(target, prop, receiver) {
    track(target, prop)
    return Reflect.get(...arguments) // get original data
  },
  set(target, key, value, receiver) {
    trigger(target, key)
    return Reflect.set(...arguments)
  }
})

const obj = reactive({
  hello: 'world'
})

console.log(obj.hello) // `track()` get called
obj.hello = 'vue' // `trigger()` get called
```

Effect

Reactivity

- track
  - track which effect called it
- trigger
  - trigger the effect bind to it
- effect
  - call the function and enable collecting

```
const targetMap = new WeakMap()

export const track = (target, key) => {
  if (tacking && activeEffect)
    targetMap.get(target).key(key).push(activeEffect)
}

export const trigger = (target, key) => {
  targetMap.get(target).key(key).forEach(effect => effect())
}

export const effect = (fn) => {
  let effect = function() { fn() }
  enableTracking()
  activeEffect = effect
  fn()
  resetTracking()
  activeEffect = undefined
}
```

Computed

Reactivity

- Computed & Watch
  - Based on Effect
- Extend Reading

```
const computed = (getter) => {
  let value
  let dirty = true

  const runner = effect(getter, {
    lazy: true,
    scheduler() {
      dirty = true // deps changed
    }
  })

  return {
    get value() {
      if (dirty) {
        value = runner() // re-evaluate
        dirty = false
      }
      return value
    }
  }
}
```

Composition

- Based on Reactivity System
- Vue's lifecycles (onMounted, onUnmounted, etc.)
- Effects auto collection & disposal on component unmount
- Reusable logics
  - Cross components
  - Cross application (Composable libraries)

Reactivity

@vue/reactivity

`ref`

`reactive`

`computed`

`effect`

Composition

@vue/runtime-core

`watch`

`setup`

`onMounted (lifecycles)`

`getCurrentInstance`

Reactivity

@vue/reactivity

```
ref

reactive

computed

effect
```

Could be used Standalone

Composition

@vue/runtime-core

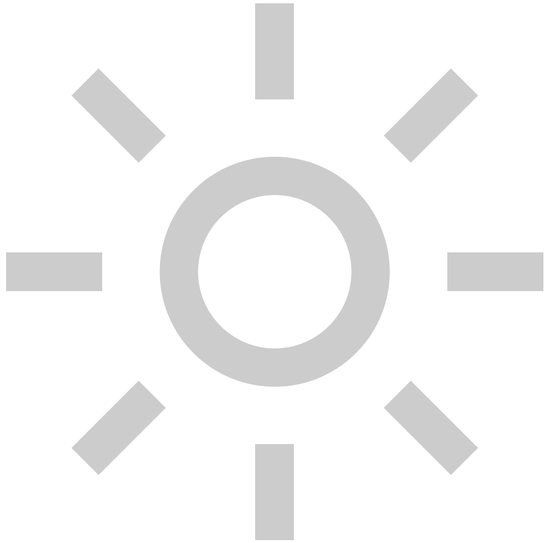
```
watch

setup

onMounted (lifecycles)

getCurrentInstance
```

Composition API Example

Case Study

Dark mode

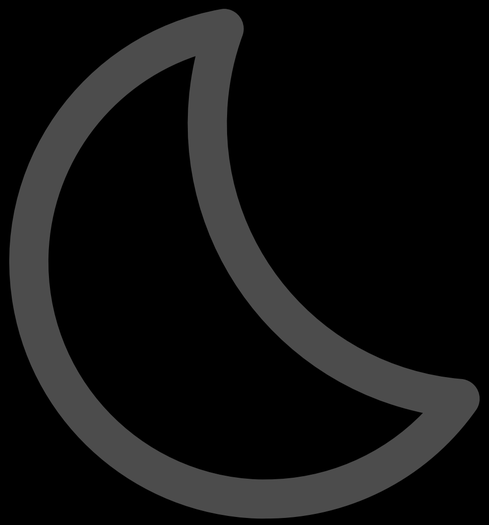
Case study



- Default to match with system preference
- Manually configurable
- Persistent
- Run code on changing

Dark mode

Case study



- Default to match with system preference
- Manually configurable
- Persistent
- Run code on changing

Basic

```
<template>
  <div :class='{dark}'>
    <button @click='toggleDark'>Toggle</button>
  </div>
</template>
```

Basic

Options API

```
<script>
export default {
  data() {
    return {
      dark: false
    }
  },
  methods: {
    toggleDark() {
      this.dark = !this.dark
    }
  }
}
</script>
```

Composition API

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

export default {
  setup() {
    const dark = ref(false)

    return {
      dark,
      toggleDark() {
        dark.value = !dark.value
      }
    }
  }
}
</script>
```

Add Media Query

Options API

```vue
<script>
export default {
  data() {
    return {
      dark: false,
      media: window.matchMedia('(prefers-color-scheme: dark)')
    }
  },
  methods: {
    toggleDark() {
      this.dark = !this.dark
    },
    update() {
      this.dark = this.media.matches
    }
  },
  created() {
    this.media.addEventListener('change', this.update)
    this.update()
  },
  destroyed() {
    this.media.removeEventListener('change', this.update)
  }
}
</script>
```

Composition API

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

export default {
  setup() {
    const media = window.matchMedia('(prefers-color-scheme: dark)')
    const dark = ref(media.matches)

    const update = () => dark.value = media.matches

    media.addEventListener('change', update)

    onUnmounted(() => {
      media.removeEventListener('change', update)
    })

    return {
      dark,
      toggleDark() {
        dark.value = !dark.value
      }
    }
  }
}
</script>
```

```vue
<script>
export default {
  data() {
    return {
      dark: false,
      media: window.matchMedia('(prefers-color-scheme: dark)'),
      setting: localStorage.getItem('setting-dark') || 'auto'
    }
  },
  methods: {
    toggleDark() {
      this.setting = this.setting === 'dark' ? 'light' : 'dark'
    },
    update() {
      if (this.setting === 'auto')
        this.dark = this.media.matches
      else
        this.dark = this.setting === 'dark'
    }
  },
  watch: {
    setting(newValue) {
      localStorage.setItem('setting-dark', newValue)
      this.update()
    }
  },
  created() {
    this.media.addEventListener('change', this.update)
    this.update()
  },
  destroyed() {
    this.media.removeEventListener('change', this.update)
  }
}
</script>
```

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

export default {
  setup() {
    const media = window.matchMedia('(prefers-color-scheme: dark)')
    const dark = ref(media.matches)
    const setting = ref(localStorage.getItem('setting-dark') || 'auto')

    const update = () => {
      if (setting.v === 'auto')
        dark.value = media.matches
      else
        dark.value = (setting.value === 'dark')
    }

    watch(setting, () => {
      localStorage.setItem('setting-dark', setting.value)
      this.update()
    })

    media.addEventListener('change', update)

    onUnmounted(() => {
      media.removeEventListener('change', update)
    })

    return {
      dark,
      toggleDark() {
        setting.value = setting.value === 'dark' ? 'light' : 'dark'
      }
    }
  }
}
</script>
```

```vue
<script>
export default {
  data() {
    return {
      dark: false,
      media: window.matchMedia('(prefers-color-scheme: dark)'),
      setting: localStorage.getItem('setting-dark') || 'auto'
    }
  },
  methods: {
    toggleDark() {
      this.setting = this.setting === 'dark' ? 'light' : 'dark'
    },
    update() {
      if (this.setting === 'auto')
        this.dark = this.media.matches
      else
        this.dark = this.setting === 'dark
    }
  },
  watch: {
    setting(newValue) {
      localStorage.setItem('setting-dark', newValue)
      this.update()
    }
  },
  created() {
    this.media.addEventListener('change', this.update)
    this.update()
  },
  destroyed() {
    this.media.removeEventListener('change', this.update)
  }
}
</script>
```

Bad Smell !

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

export default {
  setup() {
    const media = window.matchMedia('(prefers-color-scheme: dark)')
    const dark = ref(media.matches)
    const setting = ref(localStorage.getItem('setting-dark') || 'auto')

    const update = () => {
      if (setting.v === 'auto')
        dark.value = media.matches
      else
        dark.value = (setting.value === 'dark')
    }

    watch(setting, (setting) => {
      localStorage.setItem('setting-dark', setting.value)
      update()
    })

    media.addEventListener('change', update)

    onUnmounted(() => {
      media.removeEventListener('change', update)
    })

    return {
      dark,
      toggleDark() {
        setting.value = setting.value === 'dark' ? 'light' : 'dark'
      }
    }
  }
}
</script>
```

Reuse the logics

Options API

- You can, but not ideal
- Mixin
- Renderless Component
- Vuex

Composition API

Copy & Paste!

```vue
<script>
import { useDark } from './utils'

export default {
  setup() {
    const { dark, toggleDark } = useDark()
    // other logic

    return {
      dark,
      toggleDark
    }
  }
}
</script>
```

Reuse more

One problem at a time

```ts
export function useDark() {
  const system = usePreferDark()
  const setting = useLocalStorage('setting-dark', 'auto')

  const dark = computed({
    get() {
      return setting.value === 'auto'
        ? system.value
        : setting.value === 'dark'
    },
    set(v) {
      if (v === system.value)
        setting.value = 'auto'
      else
        setting.value = v ? 'dark' : 'light'
    },
  })

  return dark
}
```

```ts
export function usePreferDark() {
  const media = window.matchMedia('(prefers-color-scheme: dark)')
  const dark = ref(media.matches)

  const update = () => dark.value = media.matches

  media.addEventListener('change', update)
  onUnmounted(() => {
    media.removeEventListener('change', update)
  })

  return dark
}
```

```ts
export function useLocalStorage(key, defaultValue) {
  const data = ref(localStorage.getItem(key) ?? defaultValue)

  watch(data, () => localStorage.setItem(key, data.value))

  return data
}
```

Reuse more

One problem at a time

```ts
export function useDark() {
  const system = usePreferDark()
  const setting = useLocalStorage('setting-dark', 'auto')

  const dark = computed({
    get() {
      return setting.value === 'auto'
        ? system.value
        : setting.value === 'dark'
    },
    set(v) {
      if (v === system.value)
        setting.value = 'auto'
      else
        setting.value = v ? 'dark' : 'light'
    },
  })

  return dark
}
```

```ts
export function usePreferDark() {
  const media = window.matchMedia('(prefers-color-scheme: dark)')
  const dark = ref(media.matches)

  const update = () => dark.value = media.matches

  media.addEventListener('change', update)
  onUnmounted(() => {
    media.removeEventListener('change', update)
  })

  return dark
}
```

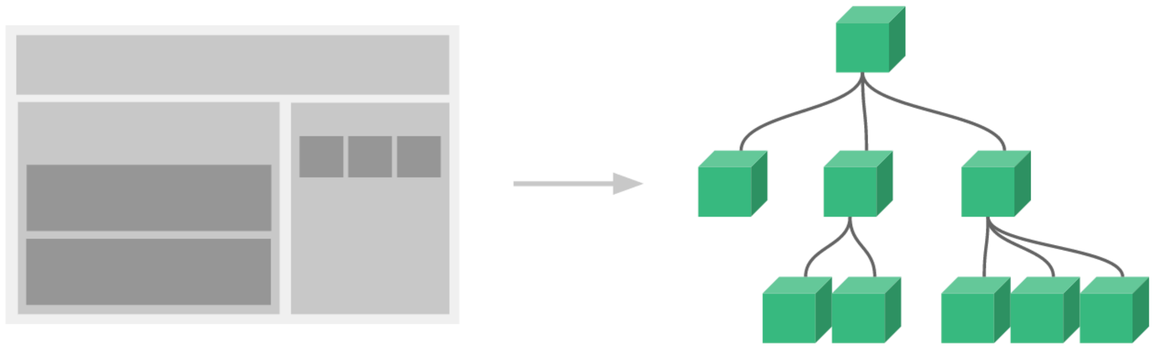
auto update & auto dispose

```ts
export function useLocalStorage(key, defaultValue) {
  const data = ref(localStorage.getItem(key) ?? defaultValue)

  watch(data, () => localStorage.setItem(key, data.value))

  return data
}
```

auto save

"Logical Components"

Logical Components



UI Components

Props → UI 🔄 State → Events

UI Components

(Reactive) Arguments → (Lifecycles) → Reactive data (ref, reactive, etc.)

Composable Libraries

Works for Vue 2 & 3 that you can already use

VueUse

Fine-grained Web API and Utilities

- useEventListener
- useMouse
- useStorage
- useMediaQuery
- useDebounce
- useWebWorkerFn
- ...

vue-composable

Useful high-level tools by @pikax

- useI18n
- useValidation
- useBreakpoints
- useDateFormat
- useUndo
- useSharedRef
- ...

Composition API Ecosystem

DevTools

6.0.0-beta-2 / @Akryum

- Vue 3 Support
- Timeline
- vue-composable by @pikax

```
import { useDevtoolsInspector } from 'vue-composable'

const counter = ref(0)

useDevtoolsInspector({ counter })
```

SFC Improvements

<script setup>

```vue
<script>
import { ref, watchEffect, defineComponent } from 'vue'
import Foo from './Foo.vue'
import { useDark } from './utils'
import { store } from './store'

export default defineComponent({
  components: { Foo },
  setup() {
    const input = ref('')
    const dark = useDark()

    /* Other logics */
    watchEffect(() => console.log('input: ', input.value))

    return { store, input, isDark }
  }
})
</script>
```

```vue
<script setup>
import { ref, watchEffect } from 'vue'
import { useDark } from './utils'

export { default as Foo } from './Foo.vue'
export { store } from './store'

export const input = ref('')
export const dark = useDark()

/* Other logics */
watchEffect(() => console.log('input: ', input.value))
</script>
```

Register Component

Export global state

Declaration & exporting in one line

Vue 2 & 3 Isomorphic

From the view of library maintainers

Isomorphic for Vue 2 & 3

```
// vue 3
import { ref, reactive } from 'vue'

// vue 2
import Vue from 'vue'
import plugin, { ref, reactive } from '@vue/composition-api'

Vue.use(plugin)
```

Isomorphic for Vue 2 & 3

- Solutions
  - Maintaining two version (publish with two npm tags)
  - Write a script to replace the API importing on build (single codebase, but two tags still)

Vue Demi

Make Universal Library for Vue 2 & 3

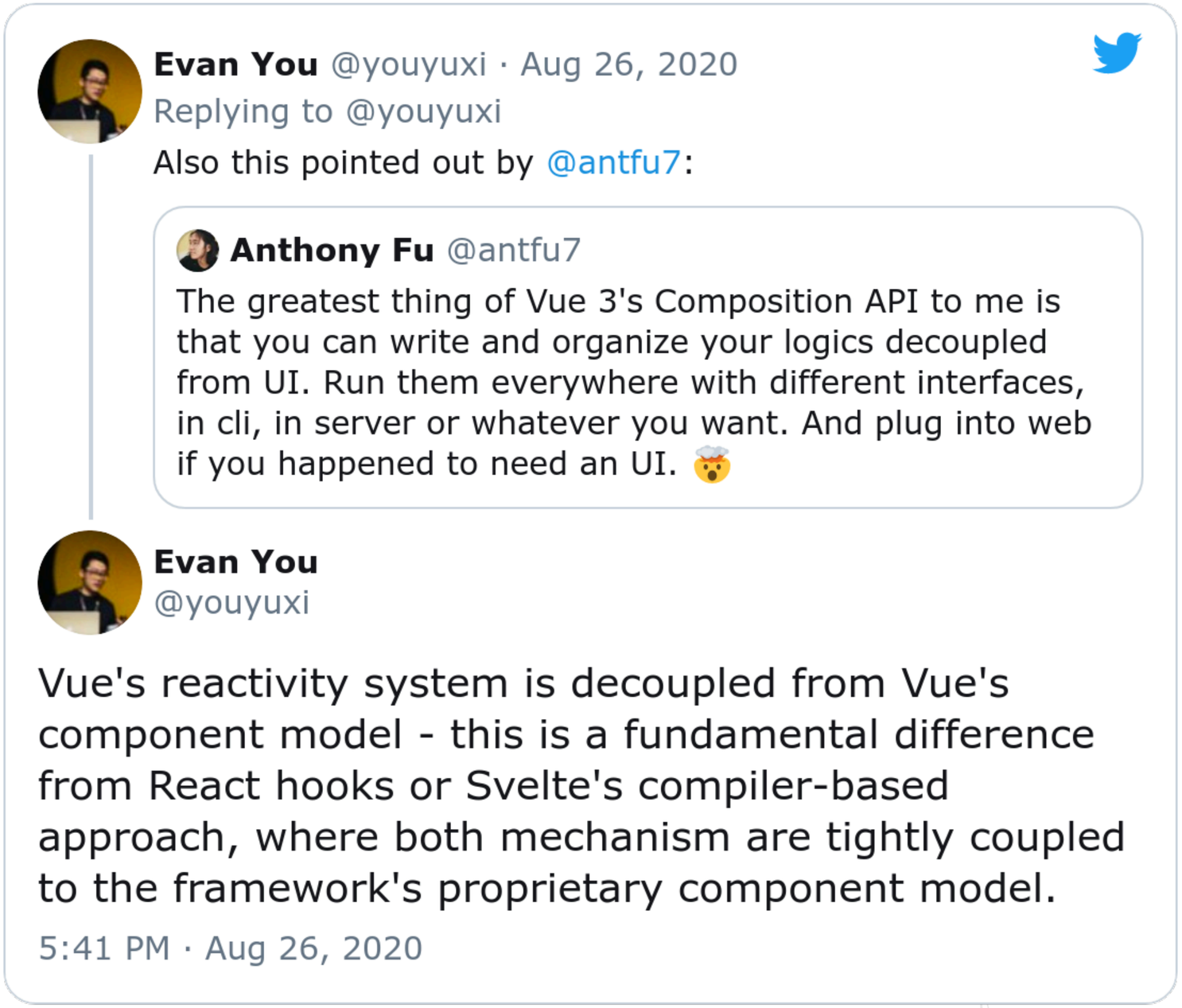
```
import { ref, reactive } from 'vue-demi'
```

- Import all APIs from `vue-demi`
- Detecting Vue's version via `postinstall`
- Redirecting to corresponding packages
- One npm package for Vue 2 & 3
- Unit Tests by @pikax

- Libraries using `vue-demi`
  - `VueUse`
  - `@vue/apollo-composable`
  - `vuelidate`
  - `vue-use-infinite-scroll`

@vue/reactivity

Flexible & Powerful Reactivity System



UI Decoupled

Decoupled from UI

- Can be used in different environment
- Or even build your own framework with ease

vue + htm

```
<div id="app"></div>

<script type="module">
import { createApp, reactive, h } from "https://unpkg.com/vue@next/dist/vue.runtime.esm-browser.js"
import htm from "https://unpkg.com/htm?module"

const html = htm.bind(h)

createApp({
  setup() {
    const state = reactive({ count: 0 })
    const increase = () => { state.count++ }

    return () => html`
      <p>Hello World</p>
      <p>${state.count}</p>
      <button onClick=${increase}>increase</button>
    `
  }
}).mount("#app")
</script>
```

vue/reactivity + lit-html

```
<div id="app"></div>

<script type="module">
import { html, render } from 'https://unpkg.com/lit-html?module'
import { reactive, effect } from "https://unpkg.com/@vue/reactivity/dist/reactivity.esm-browser.js"

const Component = () => {
  const state = reactive({ count: 0 })
  const increase = () => { state.count++ }

  return () => html`
    <p>Hello World</p>
    <p>${state.count}</p>
    <button @click=${increase}>increase</button>
  `
}

function mount(comp, target) {
  const template = comp()
  effect(() => render(template(), target))
}

mount(Component, document.querySelector('#app'))
</script>
```

reactivue (PoC)

Composition API in React

- Only rely on @vue/reactivity
- React's lifecycles in Vue style
  - onUpdated
  - onUnmounted
  - ...
- Reusable Vue libraries
  - VueUse
  - pinia

```
import React from 'React'
import { useSetup, ref, computed } from 'reactivue'

function MyCounter(props) {
  const { counter, doubled, inc } = useSetup(
    (props) => {
      const counter = ref(props.value)
      const doubled = computed(() => counter.value * 2)
      const inc = () => counter.value += 1

      return { counter, doubled, inc }
    },
    props
  )

  return (
    <div>
      <div>{counter} x 2 = {doubled}</div>
      <button onClick={inc}>Increase</button>
    </div>
  )
}

ReactDOM.render(<MyCounter value={10}>, el)
```

@vue-reactivity

Exploration on @vue/reactivity

/watch

The missing watch in @vue/reactivity

/scope

Effect auto collecting

/lifecycle (wip)

Lifecycle hooks framework

/fs (wip)

Reactive file system

/bridge (wip)

State syncing across client and server

...

```ts
import { ref, reactive, computed } from '@vue/reactivity'
import { watch, watchEffect } from '@vue-reactivity/watch'

const count = ref(1)

const stopWatch = watch(
  count,
  (newValue) => {
    console.log(`Count: ${newValue}`)
  }
)

count.value += 1
// Count: 2

stopWatch()
```

@vue-reactivity

Exploration on @vue/reactivity

/watch

The missing watch in @vue/reactivity

/scope

Effect auto collecting

/lifecycle (wip)

Lifecycle hooks framework

/fs (wip)

Reactive file system

/bridge (wip)

State syncing across client and server

...

```ts
import {
  effectScope,
  ref,
  computed,
  watch,
} from '@vue-reactivity/scope'

const counter = ref(0)

const stop = effectScope(() => {
  const doubled = computed(() => counter.value * 2)

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

  watchEffect(() => console.log('Count: ', double.value))
})

// to dispose all effects
stop()
```

@vue-reactivity

Exploration on @vue/reactivity

/watch

The missing watch in @vue/reactivity

/scope

Effect auto collecting

/lifecycle (wip)

Lifecycle hooks framework

/fs (wip)

Reactive file system

/bridge (wip)

State syncing across client and server

...

```ts
import { effect } from '@vue/reactivity'
import { useJSON } from '@vue-reactivity/fs'

const data = useJSON('data.json')

// log on file changes
effect(() => {
  console.log(data.value)
})

// write back to file
data.value = { foo: 'bar' }
data.value.hello = 'world'
```

The Future?

References

- Vue.js 3.0 Docs
- Official Repos
  - Vue 3.0
  - @vue/composition-api (plugin for Vue 2)
  - DevTools
- Third-party Libraries
  - VueUse
  - vue-demi
  - vue-composable
  - swrv
  - Vue Reactivity
- Experiments
  - reactivue

Thanks!

mail me at **hi@antfu.me**