Vue 3.0
Things I'm Totally Hooked On

Damian Dulisz

GitHub: @shentao

Twitter: @damiandulisz

Vue.js Core Team

news.vuejs.org

Vue.js News

Submit Search Archives About

#140 Tuesday, May 7, 2019

Vuex v3.1.1 released. Vue-CLI v4.0-alpha is here; Managing state with Apollo and more!

Hello!

Whoa, there is so much content this week we had a really hard time picking up the articles (some of them might show up next week).

First of all, Vuex got a new release which will definitely make the NativeScript devs happy, since now it supports debugging with the remote Vue Devtools. 🎉

The Vue-CLI v4.0.0-alpha has also been released with several meaningful changes like bumping the versions on webpack-chain, core-js, ESLint, and workbox in the PWA plugin. Take it for a spin and let us know of any issues.

Also, Natalia has written an amazing article on managing state when using Apollo and is probably a must read those of you considering using Apollo in your project.

Cheers,

— Damian Dulisz

PLAY THE PODCAST

The Official Vue News

We exist to provide Vue developers with the latest news and tutorials to stay up-to-date with their technology.

Get a weekly email

Email address SUBSCRIBE

Previously known as Vue.js Newsletter

Subscribe to the podcast

APPLE PODCASTS ANDROID

STITCHER RSS

#140 | 2019-05-07 Vuex v3.1.1 released. Vue-CLI v4.0-alpha is here; Managing state with Apollo and more!

Open Source Work

Vue-Multiselect

Universal select/multiselect/tagging component for Vue.js

Vuelidate

Simple, lightweight model-based validation for Vue.js

with Paweł Grabarz @frizi

Vue-Global-Events

Register global events as a component

with Eduardo San Martin Morote @posva

Lead Frontend Engineer

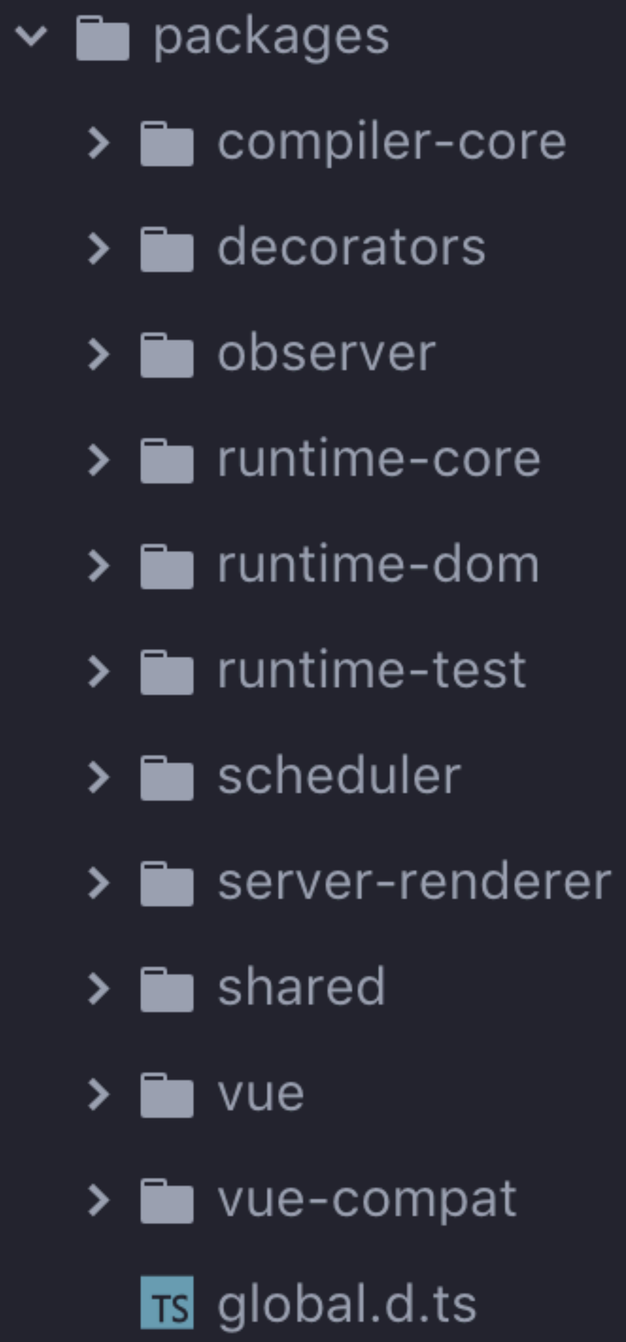
@Coursedog

Vue.js 3.0

Upcoming Changes

Rewritten from the ground up

- Cleaner and more maintainable architecture
- Internal functionalities broken into individual packages
- Easier contributions



Rewritten from the ground up

Extendability:

- Custom Renderers
- Custom Reactivity Engines

NativeScript

Vue's Global API Is Tree-Shakeable

- Works with ES modules builds
- Done automatically by the bundler
- The UMD bundle stays the same

Example:

If you don't use `<transition>` or `<keep-alive>` built-in components, `v-show` directives or APIs like `nextTick()`, they won't be included in your bundle.

Vue’s Global API Is Tree-Shakeable

New Core Runtime size:

~10kb gzipped

Current v2.x is ~20kb

Written in TypeScript

- 1st class support for TypeScript users
- Better Type inferences
- Easier library maintenance

Written in TypeScript

- 1st class support for TypeScript users
- Better Type inferences
- Easier library maintenance
- **Using TypeScript in your application is completely optional**

New Reactivity Tracking Engine

New Reactivity Tracking Engine

```
<template>
  <div>{{ firstName }}</div>
</template>

<script>
export default {
  data() {
    return {
      firstName: 'Damian',
      lastName: 'Dulisz'
    }
  },
  // ...
}
</script>
```

New Reactivity Tracking Engine

Vue 2.x

```
// Setting an array item by index
Vue.set(this.myArray, index, newValue)

// Adding a new property to an object
Vue.set(this.myObject, key, value)

// Deleting a property from an object
Vue.delete(this.myObject, key)
```

New Reactivity Tracking Engine

- Moving to a **Proxy-based** observer implementation
- Support for Map, Set, WeakMap and WeakSet

Vue 2.x

```
// Setting an array item by index
Vue.set(this.myArray, index, newValue)

// Adding a new property to an object
Vue.set(this.myObject, key, value)

// Deleting a property from an object
Vue.delete(this.myObject, key)
```

New Reactivity Tracking Engine

- Moving to a **Proxy-based** observer implementation
- Support for Map, Set, WeakMap and WeakSet

Vue 3.0

```
// Setting an array item by index
this.myArray[index] = newValue

// Adding a new property to an object
this.myObject[key] = value

// Deleting a property from an object
delete this.myObject[key]
```

No need to worry about reactivity caveats 🎉

New Reactivity Tracking Engine

- Moving to a **Proxy-based** observer implementation
- Support for Map, Set, WeakMap and WeakSet
- **Separate build for IE11 based on** `Object.defineProperty` (based on 2.x implementation)

Vue 3.0

```
// Setting an array item by index
this.myArray[index] = newValue

// Adding a new property to an object
this.myObject[key] = value

// Deleting a property from an object
delete this.myObject[key]
```

No need to worry about **reactivity caveats** 🎉

Other Quality of Life Changes

- Support for multiple root nodes
- `v-model` and `.sync` will be merged
- **Performance Improvements**

```
<template>
  <div>Node 1</div>
  <div>Node 2</div>
</template>
```

```
<InviteeForm
  v-model:name="inviteeName"
  v-model:email="inviteeEmail"
/>
```

Performance Improvements

Double the speed
Half the memory usage

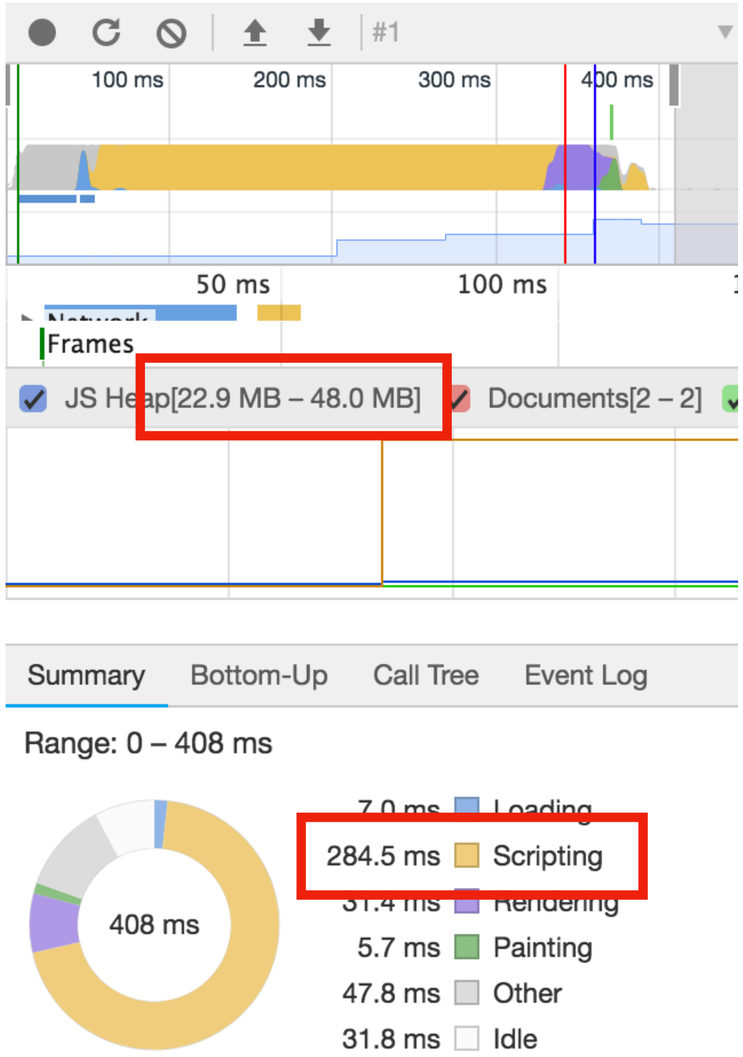
— Evan You

Double the speed
Half the memory usage

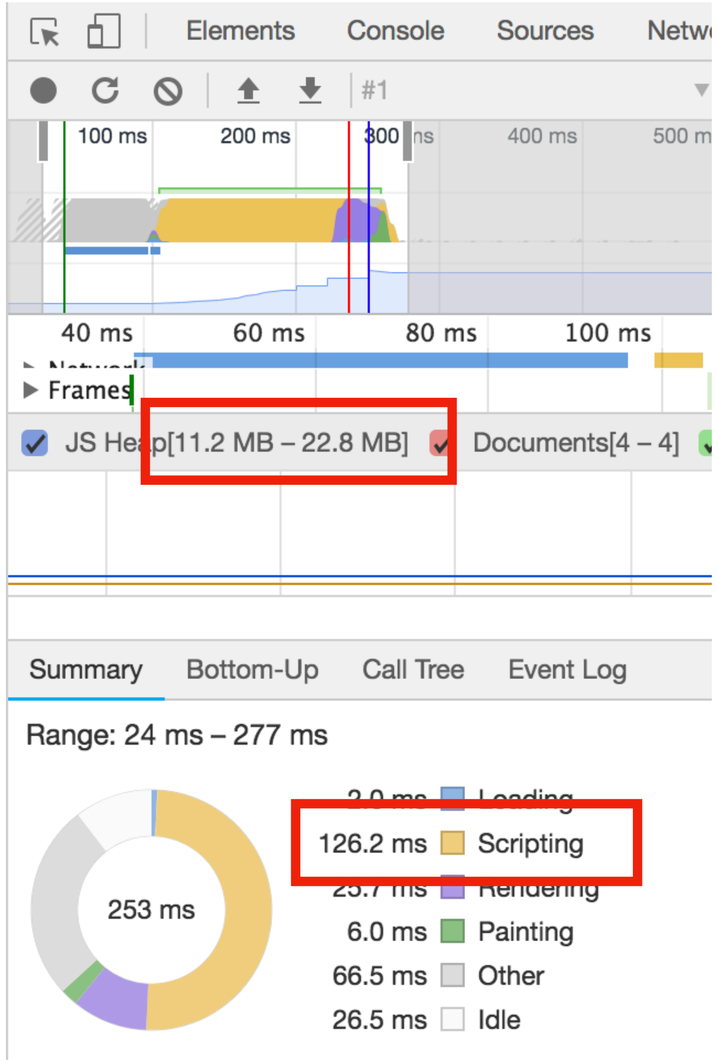
Up to 100% faster Component instance initialisation

Double the speed
Half the memory usage

Rendering 3000 stateful component instances



v2.5



v3.0-proto

Now for the *really* fancy changes

Wait.

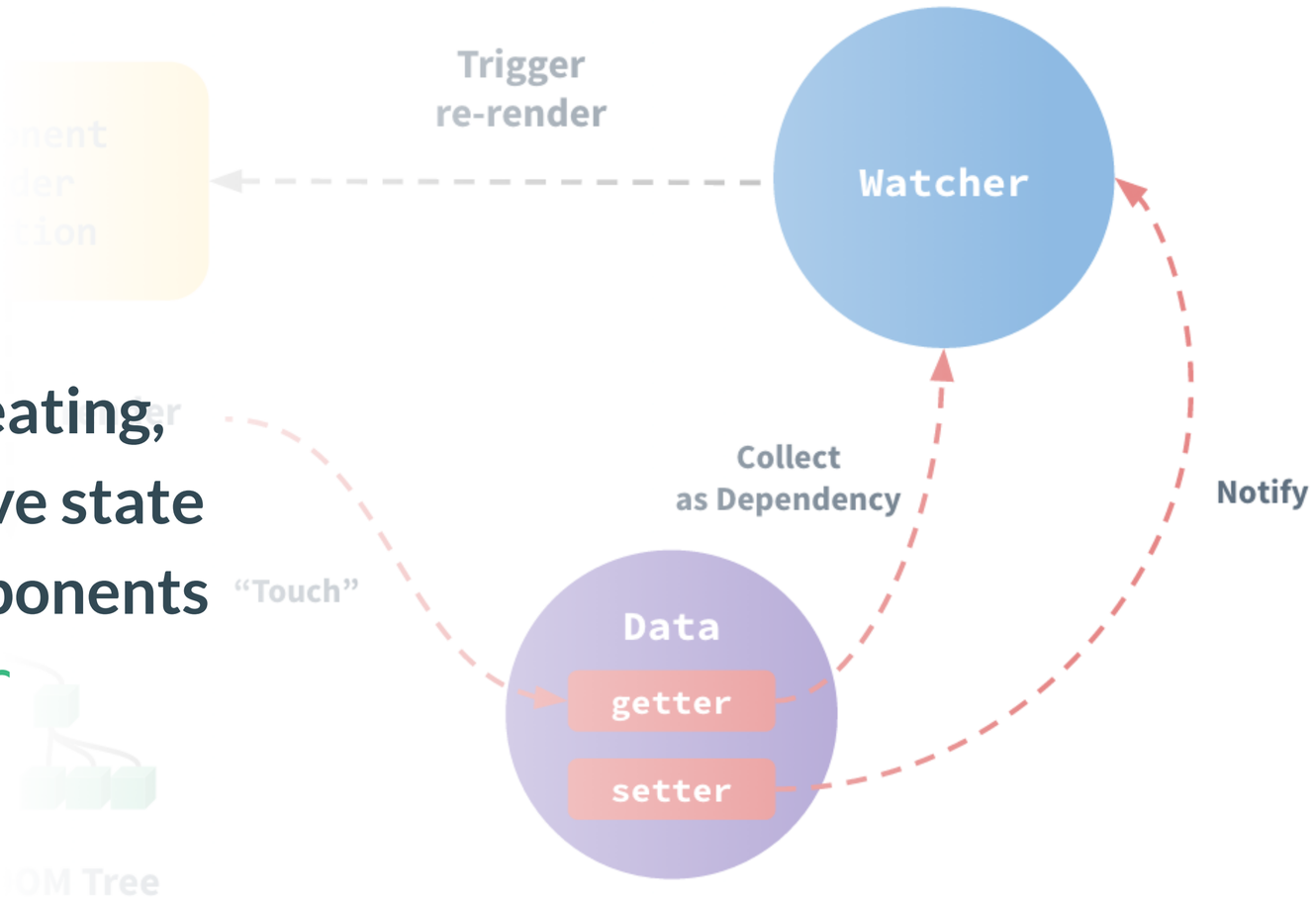
No Class API?

For creating components

We can do better

Advanced Reactivity API

- Provides standalone APIs for creating, observing and reacting to reactive state
- Can be used outside of Vue components
- Imported from `@vue/observer`



Advanced Reactivity API

```
import { state } from '@vue/observer'

// create a reactive object
const store = state({ counter: 1 })
```

Advanced Reactivity API

```
import { state, watch } from '@vue/observer'

// create a reactive object
const store = state({ counter: 1 })

// create a watcher
watch(
  // whenever counter changes
  () => store.counter,
  // execute callback
  counter => {
    console.log(`store.counter is now: ${counter}`)
  }
)
```

Advanced Reactivity API

```
import { state, value } from '@vue/observer'

// only works for objects and arrays
// that are passed by reference
const store = state({ counter: 1 })

// create an object wrapper around a primitive value
// which acts as a pointer that can be passed by reference
const counter = value(0)

// read the pointer's value
console.log(counter.value) // 0

// mutate the value
counter.value++
```

Advanced Reactivity API

```
import { value, computed } from '@vue/observer'

const counter = value(0)

// create a computed pointer
const counterPlusOne = computed(() => counter.value + 1)
```

Advanced Reactivity API

```
import { value, watch, computed } from '@vue/observer'

const counter = value(0)

// create a computed pointer
const counterPlusOne = computed(() => counter.value + 1)

// pointers can be watched directly
watch(counter, (count, oldCount) => {
  console.log(`count is: ${count}`)
})

watch(counterPlusOne, countPlusOne => {
  console.log(`count plus one is: ${countPlusOne}`)
})
```

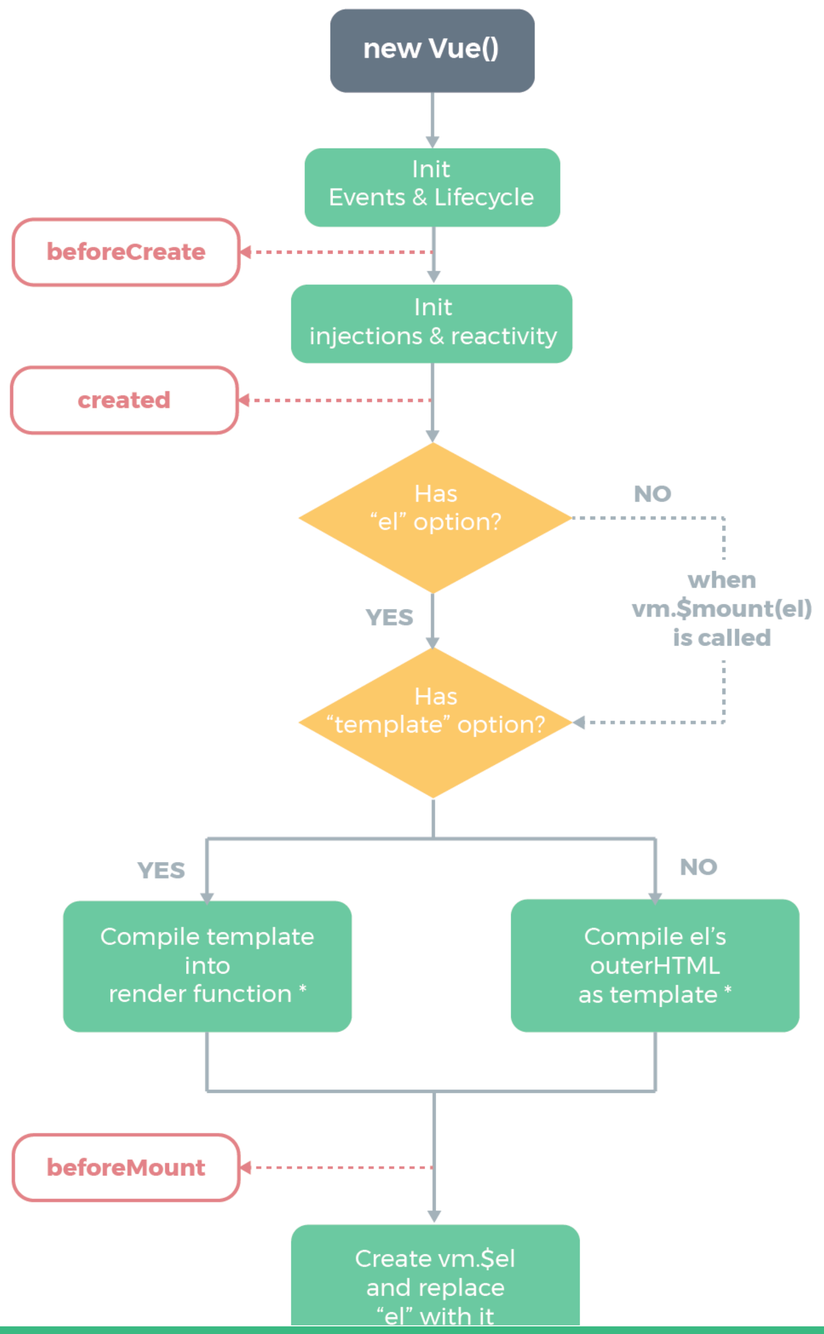
Advanced Reactivity API

Why is this exciting?

- Entirely new ways to build Vue plugins that make heavy use of reactive data like **Vuelidate**
- Easier integration with 3rd-party SDKs and libraries like **Rx.js** and **jQuery**
- Opens the way for completely new Vuex alternatives
- Vuex itself might be rewritten using this API

Dynamic Lifecycle Injection

- Allows to dynamically add and remove lifecycle hooks



Dynamic Lifecycle Injection

```
import { onMounted, onUpdated, onDestroyed } from 'vue'

// inside a Vue component
export default {
  created() {
    onMounted(() => {
      console.log('mounted')
    })

    onUpdated(() => {
      console.log('updated')
    })

    onDestroyed(() => {
      console.log('destroyed')
    })
  }
}
```

Dynamic Lifecycle Injection

```
import { onMounted, onUpdated, onDestroyed } from 'vue'

// an updated hook that fires only once
const remove = onUpdated(() => {
  // ...
  remove()
})
```

Dynamic Lifecycle Injection

```
import { onMounted, onUpdated, onDestroyed } from 'vue'

// an updated hook that fires only once
const remove = onUpdated(() => {
  // ...
  remove()
})

// target instance can be passed in via the 2nd argument
onMounted(() => { /* ... */ }, targetInstance)
```

Dynamic Lifecycle Injection

Vue 2.x

```
// inside a Vue component
export default {
  created () {
    let updates = 0

    const onUpdated = () => {
      console.log('updated', ++updates, 'times')
      if (updates > 5) {
        // Remove the hook listener
        this.$off('hook:updated', onUpdated)
      }
    }
    // private API, AVOID!
    this.$on('hook:updated', onUpdated)
  }
}
```

Dynamic Lifecycle Injection

```
<MyComponent
  @hook:update="handleChildUpdate"
/>
```

Dynamic Lifecycle Injection

This is considered a private API that might change at some point. Try not to use it.

```
<MyComponent
  @hook:update="handleChildUpdate"
/>
```

Dynamic Lifecycle Injection

This is considered a private API that might change at some point. Try not to use it.

```
<MyComponent
  @hook:update="handleChildUpdate"
/>
```

Advanced Reactivity API

+

Dynamic Lifecycle Injection

=

Advanced Reactivity API + Dynamic Lifecycle Injection = New Composition Pattern

New Composition Pattern

Existing Composition Patterns

Mixins

Higher Order Components

Scoped Slots

Existing Composition Patterns

Mixins

Scoped Slots

Known as Render Props in React ⚛️

Mixins

```
const mousePositionMixin = {
  data() {
    return {
      x: 0,
      y: 0
    }
  },
  mounted() {
    window.addEventListener('mousemove', this.update)
  },
  destroyed() {
    window.removeEventListener('mousemove', this.update)
  },
  methods: {
    update(e) {
      this.x = e.x
      this.y = e.y
    }
  }
}

// mixin usage in a component
export default {
  mixins: [mousePositionMixin],
  // ...
}
```

Mixins

Pros

- Relatively easy to use
- No additional component instances

```
const mousePositionMixin = {
  data() {
    return {
      x: 0,
      y: 0
    }
  },
  mounted() {
    window.addEventListener('mousemove', this.update)
  },
  destroyed() {
    window.removeEventListener('mousemove', this.update)
  },
  methods: {
    update(e) {
      this.x = e.x
      this.y = e.y
    }
  }
}

// mixin usage in a component
export default {
  mixins: [mousePositionMixin],
  // ...
}
```

Mixins

Pros

- Relatively easy to use
- No additional component instances

Cons

- Namespace clash
- Unclear property origin

```
const mousePositionMixin = {
  data() {
    return {
      x: 0,
      y: 0
    }
  },
  mounted() {
    window.addEventListener('mousemove', this.update)
  },
  destroyed() {
    window.removeEventListener('mousemove', this.update)
  },
  methods: {
    update(e) {
      this.x = e.x
      this.y = e.y
    }
  }
}

// mixin usage in a component
export default {
  mixins: [mousePositionMixin],
  // ...
}
```

Scoped Slots

As Renderless Components

```
export default {
  data() {
    return { x: 0, y: 0 }
  },
  mounted() {
    window.addEventListener('mousemove', this.update)
  },
  beforeDestroy() {
    window.removeEventListener('mousemove', this.update)
  },
  methods: {
    update(e) {
      this.x = e.x
      this.y = e.y
    }
  },
  render() {
    return this.$scopedSlots.default(
      { x: this.x, y: this.y }
    )
  },
}

<!-- USAGE -->
<WithMouse>
  <template v-slot="{ x, y }">
    Mouse position: x {{ x }} / y {{ y }}
  </template>
</WithMouse>
```

Scoped Slots

As Renderless Components

Pros

- Clear source of variables
- No namespace issues
- Can be composed together

```
export default {
  data() {
    return { x: 0, y: 0 }
  },
  mounted() {
    window.addEventListener('mousemove', this.update)
  },
  beforeDestroy() {
    window.removeEventListener('mousemove', this.update)
  },
  methods: {
    update(e) {
      this.x = e.x
      this.y = e.y
    }
  },
  render() {
    return this.$scopedSlots.default(
      { x: this.x, y: this.y }
    )
  },
}

<!-- USAGE -->
<WithMouse>
  <template v-slot="{ x, y }">
    Mouse position: x {{ x }} / y {{ y }}
  </template>
</WithMouse>
```

Scoped Slots

As Renderless Components

Pros

- Clear source of variables
- No namespace issues
- Can be composed together

Cons

- Extra component instance
- Access to exposed variables and methods only in the template

```
export default {
  data() {
    return { x: 0, y: 0 }
  },
  mounted() {
    window.addEventListener('mousemove', this.update)
  },
  beforeDestroy() {
    window.removeEventListener('mousemove', this.update)
  },
  methods: {
    update(e) {
      this.x = e.x
      this.y = e.y
    }
  },
  render() {
    return this.$scopedSlots.default(
      { x: this.x, y: this.y }
    )
  },
}

<!-- USAGE -->
<WithMouse>
  <template v-slot="{ x, y }">
    Mouse position: x {{ x }} / y {{ y }}
  </template>
</WithMouse>
```

New Composition Pattern

Hooks?

Composition functions

Composition Functions

```
import { onMounted, onDestroyed } from 'vue'
import { state } from '@vue/observer'

export default function useMousePosition() {
  const mousePos = state({
    x: 0,
    y: 0
  })

  const update = e => {
    mousePos.x = e.x
    mousePos.y = e.y
  }

  onMounted(() => {
    window.addEventListener('mousemove', update)
  })

  onDestroyed(() => {
    window.removeEventListener('mousemove', update)
  })

  return mousePos
}
```

```vue
import { onMounted, onDestroyed } from 'vue'
import { state } from '@vue/observer'

export default function useMousePosition() {
  const mousePos = state({
    x: 0,
    y: 0
  })

  const update = e => {
    mousePos.x = e.x
    mousePos.y = e.y
  }

  onMounted(() => {
    window.addEventListener('mousemove', update)
  })

  onDestroyed(() => {
    window.removeEventListener('mousemove', update)
  })

  return mousePos
}
```

```vue
<template>
  <div>
    {{ x, y }}
  </div>
</template>

<script>
import useMousePosition from './mousePosition'

export default {
  data() {
    const mousePos = useMousePosition()
    return {
      x: mousePos.x,
      y: mousePos.y
    }
  }
}
</script>
```

```python
import { onMounted, onDestroyed } from 'vue'
import { state } from '@vue/observer'

export default function useMousePosition() {
  const mousePos = state({
    x: 0,
    y: 0
  })

  const update = e => {
    mousePos.x = e.x
    mousePos.y = e.y
  }

  onMounted(() => {
    window.addEventListener('mousemove', update)
  })

  onDestroyed(() => {
    window.removeEventListener('mousemove', update)
  })

  return mousePos
}
```

```
<template>
  <div>
    {{ x, y }}
  </div>
</template>

<script>
import useMousePosition from './mousePosition'

export default {
  data() {
    const mousePos = useMousePosition()
    return {
      x: mousePos.x,
      y: mousePos.y
    }
  }
}
</script>
```

Composition Functions

Pros

- Relatively easy to use
- No additional component instances
- No namespace issues
- Clear origin of variables and functionality

Composition Functions

Even More Pros

- React-hooks like composability
  - Using composition functions inside other composition functions
- Differences from Hooks
  - Can be used in stateful components
  - Only called once
  - No call-order constraints

Excited?

Sadly its
all top secret

Just kidding

It’s all in the RFCs

(RFC = Request For Comment)

github.com/vuejs/rfcs

github.com/vuejs/rfcs

Learn more

Join the discussion

Keep in mind ⚠️

Everything presented here is still subject to changes before 3.0 ships.

When Vue 3.0 will be ready?

When Vue 3.0 will be ready?

When it’s ready.

When Vue 3.0 will be ready?

When it’s ready.

My guess would be Q4, 2019.

Thank You!

Damian Dulisz

GitHub: @shentao

Twitter: @damiandulisz

Vue.js Core Team