Option API vs Composition API:

Choosing the Right Approach for Your Team

Ben Hong

@bencodezen

A little about me

- Staff DX Engineer at Netlify
- Vue.js Core Team Member
- Nuxt Ambassador
- Vue Mastery Instructor
- Google Developer Expert

Before Vue 3, things were “simpler”…

Because there was only one way to write your components: Options API

options-api.vue

```vue
<script>
export default {
  data: () => ({
    count: 0
  }),
  computed: {
    doubleCount() {
      return this.count * 2
    }
  },
  methods: {
    incrementCount() {
      this.count++
    }
  }
}
</script>

<template>
  <h1>Standard Counter</h1>
  <p>Current Count: {{ count }}</p>
  <p>Double Count: {{ doubleCount }}</p>
  <button @click="incrementCount">+</button>
</template>
```

But with Vue 3 introducing the Composition API, it got a little more complicated…

composition-api.vue

```vue
<script>
export default {
  data: () => ({
    count: 0
  }),
  computed: {
    doubleCount() {
      return this.count * 2
    }
  },
  methods: {
    incrementCount() {
      this.count++
    }
  }
}
</script>

<template>
  <h1>Standard Counter</h1>
  <p>Current Count: {{ count }}</p>
  <p>Double Count: {{ doubleCount }}</p>
  <button @click="incrementCount">+</button>
</template>
```

composition-api.vue

```vue
<script>
export default {
  setup() {

  },
  data: () => ({
    count: 0
  }),
  computed: {
    doubleCount() {
      return this.count * 2
    }
  },
  methods: {
    incrementCount() {
      this.count++
    }
  }
}
</script>

<template>
  <h1>Standard Counter</h1>
  <p>Current Count: {{ count }}</p>
  <p>Double Count: {{ doubleCount }}</p>
```

composition-api.vue

```vue
<script>
export default {
  setup() {
    const count = 0
  },
  computed: {
    doubleCount() {
      return this.count * 2
    }
  },
  methods: {
    incrementCount() {
      this.count++
    }
  }
}
</script>

<template>
  <h1>Standard Counter</h1>
  <p>Current Count: {{ count }}</p>
  <p>Double Count: {{ doubleCount }}</p>
  <button @click="incrementCount">+</button>
</template>
```

composition-api.vue

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

export default {
  setup() {
    const count = ref(0)
  },
  computed: {
    doubleCount() {
      return this.count * 2
    }
  },
  methods: {
    incrementCount() {
      this.count++
    }
  }
}
</script>

<template>
  <h1>Standard Counter</h1>
  <p>Current Count: {{ count }}</p>
  <p>Double Count: {{ doubleCount }}</p>
  <button @click="incrementCount"></button>
```

composition-api.vue

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

export default {
  setup() {
    const count = ref(0)
    const doubleCount = computed(() => count.value * 2)
  },
  methods: {
    incrementCount() {
      this.count++
    }
  }
}
</script>

<template>
  <h1>Standard Counter</h1>
  <p>Current Count: {{ count }}</p>
  <p>Double Count: {{ doubleCount }}</p>
  <button @click="incrementCount">+</button>
</template>
```

composition-api.vue

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

export default {
  setup() {
    const count = ref(0)
    const doubleCount = computed(() => count.value * 2)
  },
  methods: {
    incrementCount() {
      this.count++
    }
  }
}
</script>

<template>
  <h1>Standard Counter</h1>
  <p>Current Count: {{ count }}</p>
  <p>Double Count: {{ doubleCount }}</p>
  <button @click="incrementCount">+</button>
</template>
```

composition-api.vue

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

export default {
  setup() {
    const count = ref(0)
    const doubleCount = computed(() => count.value * 2)
    const incrementCount = () => { count.value++ }
  }
}
</script>

<template>
  <h1>Standard Counter</h1>
  <p>Current Count: {{ count }}</p>
  <p>Double Count: {{ doubleCount }}</p>
  <button @click="incrementCount">+</button>
</template>
```

composition-api.vue

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

export default {
  setup() {
    const count = ref(0)
    const doubleCount = computed(() => count.value * 2)
    const incrementCount = () => { count.value++ }

    return {
      count,
      doubleCount,
      incrementCount
    }
  }
}
</script>

<template>
  <h1>Standard Counter</h1>
  <p>Current Count: {{ count }}</p>
  <p>Double Count: {{ doubleCount }}</p>
  <button @click="incrementCount">+</button>
</template>
```

And with Vue 3.2's release of the new script setup syntax, it adds yet another style of writing components…

script-setup.vue

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

export default {
  setup() {
    const count = ref(0)
    const doubleCount = computed(() => count.value * 2)
    const incrementCount = () => { count.value++ }

    return {
      count,
      doubleCount,
      incrementCount
    }
  }
}
</script>

<template>
  <h1>Standard Counter</h1>
  <p>Current Count: {{ count }}</p>
  <p>Double Count: {{ doubleCount }}</p>
  <button @click="incrementCount">+</button>
</template>
```

script-setup.vue

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

setup() {
  const count = ref(0)
  const doubleCount = computed(() => count.value * 2)
  const incrementCount = () => { count.value++ }

  return {
    count,
    doubleCount,
    incrementCount
  }
}
</script>

<template>
  <h1>Standard Counter</h1>
  <p>Current Count: {{ count }}</p>
  <p>Double Count: {{ doubleCount }}</p>
  <button @click="incrementCount">+</button>
</template>
```

script-setup.vue

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

const count = ref(0)
const doubleCount = computed(() => count.value * 2)
const incrementCount = () => { count.value++ }

return {
  count,
  doubleCount,
  incrementCount
}
</script>

<template>
  <h1>Standard Counter</h1>
  <p>Current Count: {{ count }}</p>
  <p>Double Count: {{ doubleCount }}</p>
  <button @click="incrementCount">+</button>
</template>
```

script-setup.vue

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

const count = ref(0)
const doubleCount = computed(() => count.value * 2)
const incrementCount = () => { count.value++ }
</script>

<template>
  <h1>Standard Counter</h1>
  <p>Current Count: {{ count }}</p>
  <p>Double Count: {{ doubleCount }}</p>
  <button @click="incrementCount">+</button>
</template>
```

So which one is the “right” one?

Approach #1:
Options API

options-api.vue

```vue
<script>
export default {
  data: () => ({
    count: 0
  }),
  computed: {
    doubleCount() {
      return this.count * 2
    }
  },
  methods: {
    incrementCount() {
      this.count++
    }
  }
}
</script>

<template>
  <h1>Standard Counter</h1>
  <p>Current Count: {{ count }}</p>
  <p>Double Count: {{ doubleCount }}</p>
  <button @click="incrementCount">+</button>
</template>
```

Approach #1:
Options API

Pros:

- Easy to learn
- Consistency
- Simplicity

Approach #1:
Options API

Pros:

- Easy to learn
- Consistency
- Simplicity

Cons:

- Opinionated
- Less flexibility
- Not as TS friendly

Approach #2:
Composition API

composition-api-script-setup.vue

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

const count = ref(0)
const doubleCount = computed(() => count.value * 2)
const incrementCount = () => { count.value++ }
</script>

<template>
  <h1>Standard Counter</h1>
  <p>Current Count: {{ count }}</p>
  <p>Double Count: {{ doubleCount }}</p>
  <button @click="incrementCount">+</button>
</template>
```

Approach #2:
Composition API

Pros:

- Flexibility
- “Just JavaScript”
- TS friendly

Approach #2:
Composition API

Pros:

- Flexibility
- “Just JavaScript”
- TS friendly

Cons:

- More flexibility
- Higher barrier of entry
- Lack of structure

Approach #3:
Options API + Composition API

options-and-composition-api.vue

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

export default {
  setup() {
    const count = ref(0)
    const doubleCount = computed(() => count.value * 2)
    const incrementCount = () => { count.value++ }

    return {
      count,
      doubleCount,
      incrementCount
    }
  }
}
</script>

<template>
  <h1>Standard Counter</h1>
  <p>Current Count: {{ count }}</p>
  <p>Double Count: {{ doubleCount }}</p>
  <button @click="incrementCount">+</button>
</template>
```

options-and-composition-api.vue

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

export default {
  setup() {
    const count = ref(0)
    const doubleCount = computed(() => count.value * 2)
    const incrementCount = () => { count.value++ }

    return {
      count,
      doubleCount,
      incrementCount
    }
  },
  computed: {
    tripleCount() {
      return this.count * 3
    }
  }
}
</script>

<template>
```

Approach #3:
Options API + Composition API

Pros:

- Structure w/ flexibility
- Progressive enhancement
- More TS friendly

Approach #3:
Options API + Composition API

Pros:

- Structure w/ flexibility
- Progressive enhancement
- More TS friendly

Cons:

- Two approaches
- More verbose
- Not as TS friendly

So which one is the “right” one?

What matters is choosing the right approach for your team.

Aspects to Consider

Aspects to Consider

- Migrating or new?
- Who is contributing?
- Plan on using TypeScript heavily?
- What does the team prefer?

Final Verdict?

Every approach is a valid one.

Every approach is a valid one regardless of what everyone else says.

Because at the end of the day, what matters are the features being shipped and whether the team enjoys working with the code base.

Thank you!

bencodezen.io

twitter.com/bencodezen

youtube.com/bencodezen

twitch.tv/bencodezen