

The background is a solid teal color with various faint, white line-art sketches overlaid. These sketches include architectural elements like a brick wall, a spiral staircase, and a building facade. There are also scientific or technical drawings, such as a circular diagram with concentric rings, a molecular structure, and a diagram of a bridge or tower. A small crescent moon is visible in the upper right corner.

Vue Fundamentals

Options API / Composition API

Peter Kassenaar –
info@kassenaar.com

Vue historically used the
'Options' API.

Since Vue 3, the
'Composition API'
is the default.

What the heck is that??

(and why should I bother?)

Traditionally (Vue 1, Vue 2)

- “Option based API”, b/c components are written as objects with key/value pairs, the *options*, defined in Vue documentation
- Code is organized by **option** and not by **functionality**
- Data, methods, computed properties are **spread all over** the component

```
<script>
  export default {
    name: "CounterStandard",
    data() {
      return {
        ...
      }
    },
    props: {
      ...
    },
    created() {
      ...
    },
    methods: {
      ...
    }
  }
</script>
```

This is **not wrong**
per se, but has
some limitations

Vue 3 – Composition API

- Currently: the **recommended** way of creating Vue Components

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

// reactive state
const count = ref(0)

// function to mutate state and trigger updates
const increment = () => {
  count.value++;
}


</script>
```

Why Composition API?

- Better **reuse of logic**
 - Create composable functions and reuse it in applications and projects
- More **flexible code organization**
 - State, functionality, data, everything has its own unique place instead of scattered all over the component
- Better **Type inference**
 - Better alignment with current TypeScript practices as a lot of companies do
- **Smaller bundle size**, less overhead
 - Better performance on the client

Vue 3 uses Composition API by default

```
1  <script setup> show component usages
2  defineProps({ props: {
3    msg: {
4      type: String,
5      required: true,
6    },
7    name: {
8      type: String,
9      required: false
10   }
11 })
12 </script>
13
14 <template>
15   <div class="greetings">
16     <h1 class="green">{{ msg }}</h1>
17     <h3>
18       You've successfully created a project with
19       <a href="https://vite.dev/" target="_blank" rel="noopener">Vite</a> +
20       <a href="https://vuejs.org/" target="_blank" rel="noopener">Vue 3</a>.
21     </h3>
22     <h2> --> My name is: {{ name }}</h2>
23   </div>
24 </template>
25
26 <style scoped> PeterKassenaar, 4-12-2024 14:12 • feat: update example to Vite
27 h1 {
28   font-weight: 500;
```

 Vue.js

Ctrl K

Docs

API


Playground


Ecosystem

About

Sponsor

Experts NEW





Composition API

setup()

- Basic Usage
- Accessing Props
- Setup Context
- Usage with Render Functions

Reactivity: Core

- ref()
- computed()
- reactive()
- readonly()
- watchEffect()
- watchPostEffect()
- watchSyncEffect()
- watch()

Reactivity: Utilities

- isRef()
- unref()
- toRef()
- toValue()
- toRefs()
- isProxy()
- isReactive()
- isReadonly()

Reactivity: Advanced

- shallowRef()
- triggerRef()
- customRef()
- shallowReactive()

Lifecycle Hooks

- onMounted()
- onUpdated()
- onUnmounted()
- onBeforeMount()
- onBeforeUpdate()
- onBeforeUnmount()
- onErrorCaptured()
- onRenderTracked()
- onRenderTriggered()
- onActivated()
- onDeactivated()
- onServerPrefetch()

Dependency Injection

<https://vuejs.org/api/>

Checkpoint

- You know the **Composition API** is the preferred way of creating new Vue applications.
- The **Options API** is not going away.
 - Existing Vue-applications can still be maintained and updated
- Use the `<script setup>` block in Composition API
- You know where to find **more information** on the Composition API.