

What is the “Composition API”?

Thus far, we used the **Options API** for building Vue apps / components.

```
{  
  data() {  
    return { name: 'Max' };  
  },  
  methods: { ... }  
}
```



This approach is absolutely fine and you can stick to it!

What is the "Composition API"?

But you **might face two main limitations / issues** when building bigger Vue apps.

Code that **belongs together logically** is **split up** across multiple options (data, methods, computed)

Re-using logic across components can be **tricky or cumbersome**

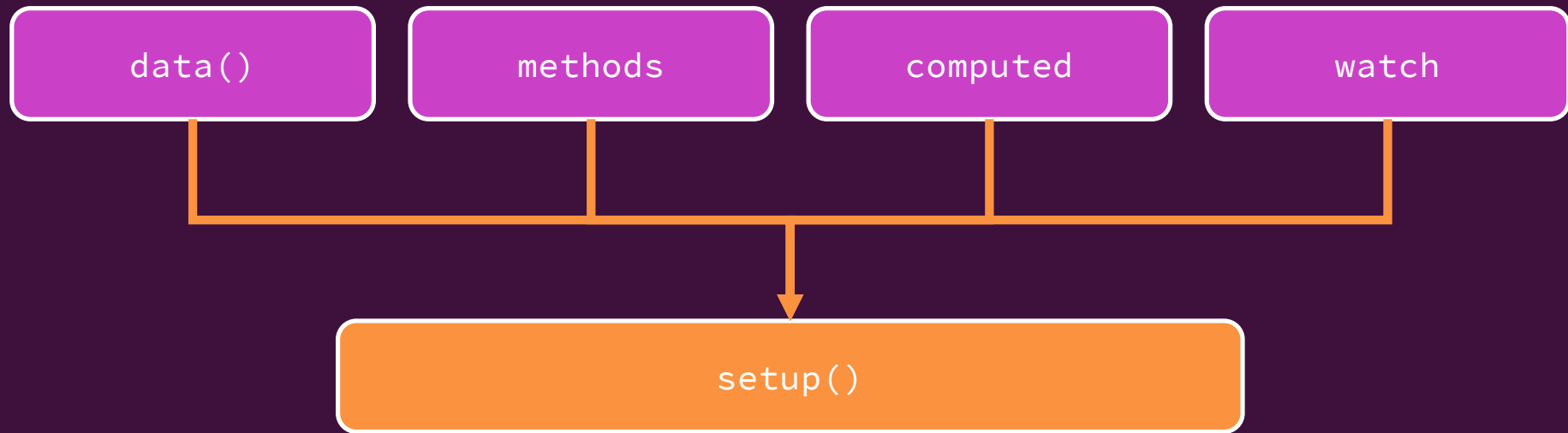
```
{  
  data() {  
    return { name: 'Max' };  
  },  
  methods: { doSth() { ... } }  
}
```



Composition API

```
{  
  setup() {  
    const name = ref('Max');  
    function doSth() { ... }  
    return { name, doSth };  
  }  
}
```

From Options API to Composition API



Options API => Composition API

Options API

`data() { ... }`

`methods: { doSmtH() { ... } }`

`computed: { val() { ... } }`

`watch: { ... }`

`provide: { ... } / inject: []`



Composition API

`ref(), reactive()`

`function doSmtH() { ... }`

`const val = computed()`

`watch(dep, (val, oldV) => {})`

`provide(key, val),
inject(key)`

Options API => Composition API: Lifecycle

Options API

beforeCreate, created

beforeMount, mounted

beforeUpdate, updated

beforeUnmount, unmounted



Composition API

Not Needed
(*setup()* replaces these hooks)

onBeforeMount, onMounted

onBeforeUpdate, onUpdated

onBeforeUnmount, onUnmounted

Composition API

What & Why

It's an **alternative** to the Options API:
It uses **setup()** to **expose** logic/ data
to the template

It's a **function-based solution** that
allows you to keep **logically related**
code together

Methods, Computed, Watchers

Methods become **regular functions**
defined in setup()

Computed properties and **watchers**
are **defined with imported functions**
(from vue)

Data & Reactivity

Data can be managed as **ref()**s
(individual values or objects) or
reactive() objects

Reactivity is a key concept – refs and
reactive objects are **reactive**, their
nested values are not

The setup() Function

The setup() function is called by Vue
when the **component is created** – it
defines data + logic for the template

setup() receives **two arguments**
(automatically): **reactive props** and
context (attrs, slots, emit())