Computed Propertie

A computed property is a reactive value that is derived from other reactive sources (like ref or reactive), and Vue automatically caches it so it is recomputed only when needed.

It's a value calculated from other data, and Vue handles when to recalculate it for you.

Need computed properties

- To derive values based on other reactive state (e.g., fullName = firstName + lastName)
- To avoid writing the same logic multiple times in your template or methods
- To optimize performance (because computed values are cached)
- To make templates cleaner and logic reusable

Types of Computed Properties : Read-only computed, Writable computed (with getter and setter),

Read-Only Computed Property

A read-only computed property is a value that is derived from one or more reactive sources (ref, reactive). Automatically updates when those sources change. Cannot be directly changed you can read it, but not write to it. Cached until a dependency changes, making it very efficient.

Think of it like a formula: you change the input, and the output updates, but you can't change the formula's result directly.

Example : Displays the full name

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

// Reactive data
const firstName = ref('Sourabha');
const lastName = ref('Joshi');

// Computed property
const fullName = computed(() => {
  return `${firstName.value} ${lastName.value}`;
});
</script>

<template>
  <h2>Computed Property Example</h2>
  <p>First Name: {{ firstName }}</p>
  <p>Last Name: {{ lastName }}</p>
  <p>Full Name (computed): {{ fullName }}</p>
</template>
```

firstName and lastName are declared using ref(), so they're reactive

fullName is declared using computed(() => ...), so it's a read-only computed

If you change firstName.value or lastName.value, fullName automatically updates

You cannot do fullName.value = 'new name' — it will throw an error, because it's read-only

Example : Total Price Calculator

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

// Reactive values
const unitPrice = ref(49.99)
const quantity = ref(1)

// Read-only computed property
const totalPrice = computed(() => {
  return (unitPrice.value * quantity.value).toFixed(2)
})
</script>

<template>
  <h2>Total Price Calculator</h2>

  <label>
    Unit Price: ${{ unitPrice }}
  </label>
  <br />

  <label>
    Quantity:
    <input type="number" v-model.number="quantity" min="1" />
  </label>

  <p>Total Price (computed): <strong>${{ totalPrice }}</strong></p>
</template>
```

Writable Computed Property

A writable computed property in Vue is, A computed value that has both a getter (to read) and a setter (to write). It is useful when you want two-way binding (e.g., input fields).It allows you to modify one or more reactive states indirectly via a single computed value.

Think of it like a wrapper around multiple values where writing to it updates the underlying state.

Sytax

```
const myValue = computed({
  get: () => {
    // return derived value
```

```
  },
  set: (newValue) => {
    // update underlying state(s)
  }
})
```

Example : full name with two way binding

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

// Reactive state
const firstName = ref('Sourabha')
const lastName = ref('Joshi')

// Writable computed property
const fullName = computed({
  get: () => `${firstName.value} ${lastName.value}`,
  set: (newValue) => {
    const parts = newValue.split(' ')
    firstName.value = parts[0] || ''
    lastName.value = parts[1] || ''
  }
})
</script>

<template>
  <h2>Writable Computed Property Example</h2>

  <label>
    First Name: <input v-model="firstName" />
  </label>
  <br />

  <label>
    Last Name: <input v-model="lastName" />
  </label>
  <br />

  <label>
    Full Name (2-way binding): <input v-model="fullName" />
  </label>

  <p>First: {{ firstName }}</p>
  <p>Last: {{ lastName }}</p>
  <p>Full: {{ fullName }}</p>
</template>
```

Example : Temperature Converter (Celsius to Fahrenheit)

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

// Reactive value in Celsius
const celsius = ref(0)

// Writable computed for Fahrenheit
const fahrenheit = computed({
  get: () => (celsius.value * 9) / 5 + 32,
  set: (newFahrenheit) => {
    celsius.value = ((newFahrenheit - 32) * 5) / 9
  }
})
</script>

<template>
  <h2>Temperature Converter</h2>

  <label>
    Celsius:
    <input type="number" v-model.number="celsius" /> °C
  </label>

  <br />

  <label>
    Fahrenheit:
    <input type="number" v-model.number="fahrenheit" /> °F
  </label>

  <p>Converted Result: {{ celsius }}°C = {{ fahrenheit }}°F</p>
</template>
```

Getting the Previous Value in Computed (Vue 3.4+)

In Vue 3.4+, the getter of a computed() function receives the previously returned value as its first argument. You can now compare with or reuse the previous computed result. No need for extra ref or watch logic.

Example : read only Computed Example

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

const count = ref(2)

// Computed tracks previous return value
const alwaysSmall = computed((previous) => {
  if (count.value <= 3) {
    return count.value
  } else {
    return previous // return the last valid value
```

```
  }
})
</script>

<template>
  <h2>Previous Value in Computed</h2>

  <button @click="count++">Increment</button>
  <p>Count: {{ count }}</p>
  <p>Always Small (<=3): {{ alwaysSmall }}</p>
</template>
```

Example : Writable Computed Example with Previous

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

const count = ref(2)

const alwaysSmall = computed({
  get(previous) {
    return count.value <= 3 ? count.value : previous
  },
  set(newValue) {
    count.value = newValue * 2
  }
})
</script>

<template>
  <h2>Writable Computed with Previous</h2>

  <input v-model.number="alwaysSmall" />
  <p>Count: {{ count }}</p>
  <p>Always Small (<=3 logic): {{ alwaysSmall }}</p>
</template>
```