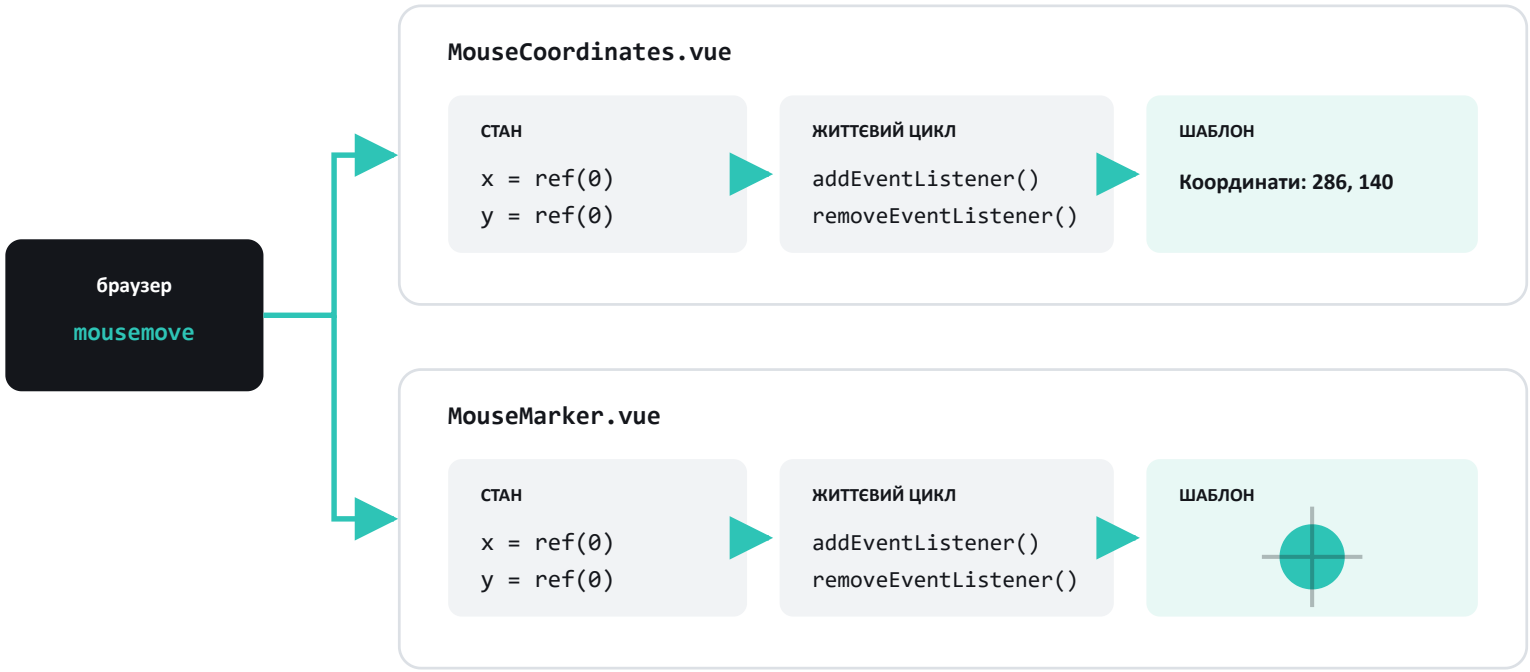
Сучасні каркаси web-додатків

Проєктування composables у Vue

Логіка зі станом, життєвий цикл і повторне використання

Лекція 5 · Лісняк Андрій Олександрович

Одна логіка в двох компонентах



> Повторне використання вимагає копіювати стан, підписку та очищення.

Vue.js — Composables

Проєктування composables у Vue

Відстеження координат у компоненті

src/MouseTracker.vue · рядки 1–10

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

const x = ref(0)
const y = ref(0)

function update(event) {
  x.value = event.pageX
  y.value = event.pageY
}
```

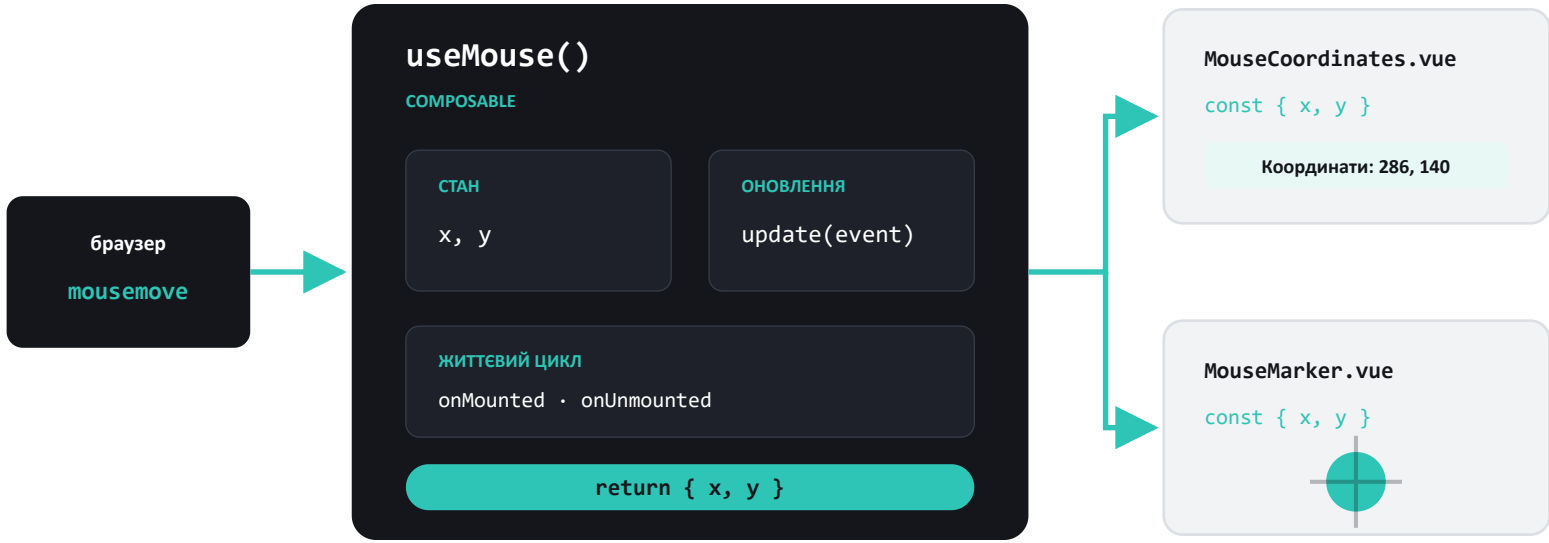
src/MouseTracker.vue · рядки 12–22

```vue
onMounted(() => {
  window.addEventListener('mousemove', update)
})
onUnmounted(() => {
  window.removeEventListener('mousemove', update)
})
</script>

<template>
  <p>Координати: {{ x }}, {{ y }}</p>
</template>
```

```
npm install && npm run dev
```

Composable відокремлює механізм



> `useMouse()` керує станом і підпискою, а компоненти визначають відображення.

Vue.js — Composables

Проєктування composables у Vue

useMouse і його використання

src/composables/useMouse.js

```
import { ref, onMounted, onUnmounted } from 'vue'
export function useMouse() {
  const x = ref(0)
  const y = ref(0)
  function update(event) {
    x.value = event.pageX
    y.value = event.pageY
  }
  onMounted(() => {
    window.addEventListener('mousemove', update)
  })
  onUnmounted(() => {
    window.removeEventListener('mousemove', update)
  })

  return { x, y }
}
```

src/MouseTrackerComposable.vue

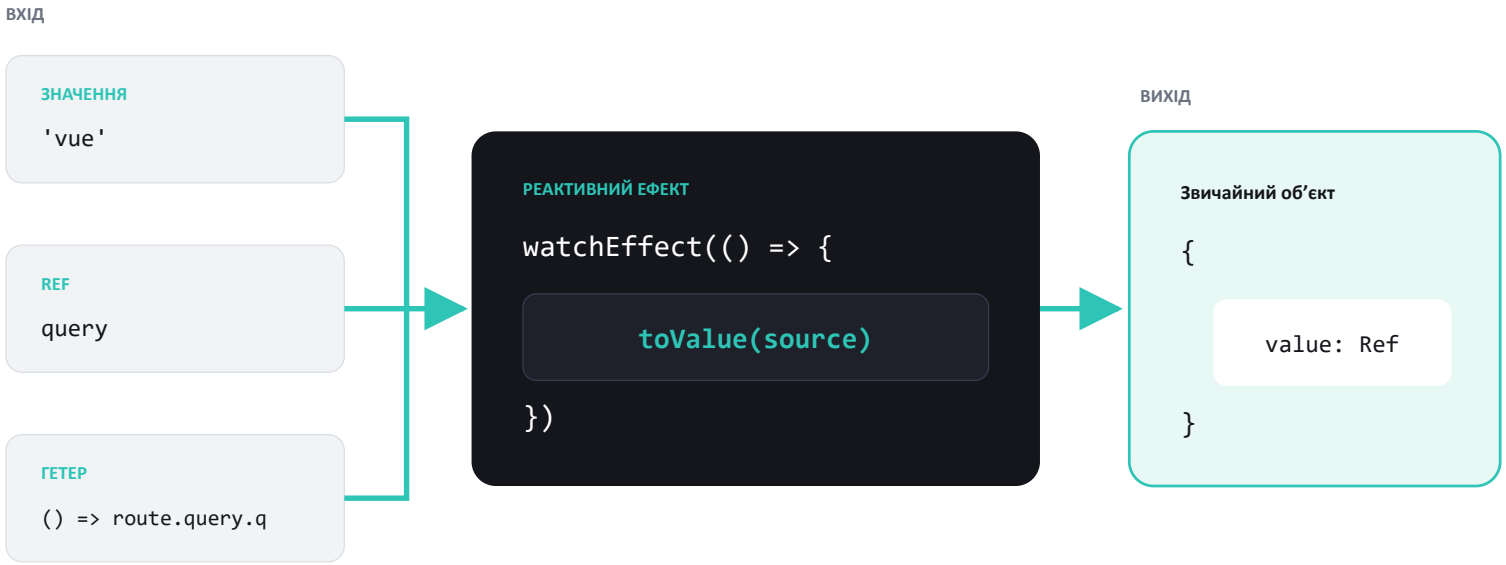
```
<script setup>
import { useMouse } from './composables/useMouse.js'

const { x, y } = useMouse()
</script>

<template>
  <p>Координати: {{ x }}, {{ y }}</p>
</template>
```

```
npm install && npm run dev
```

Контракт входу та виходу

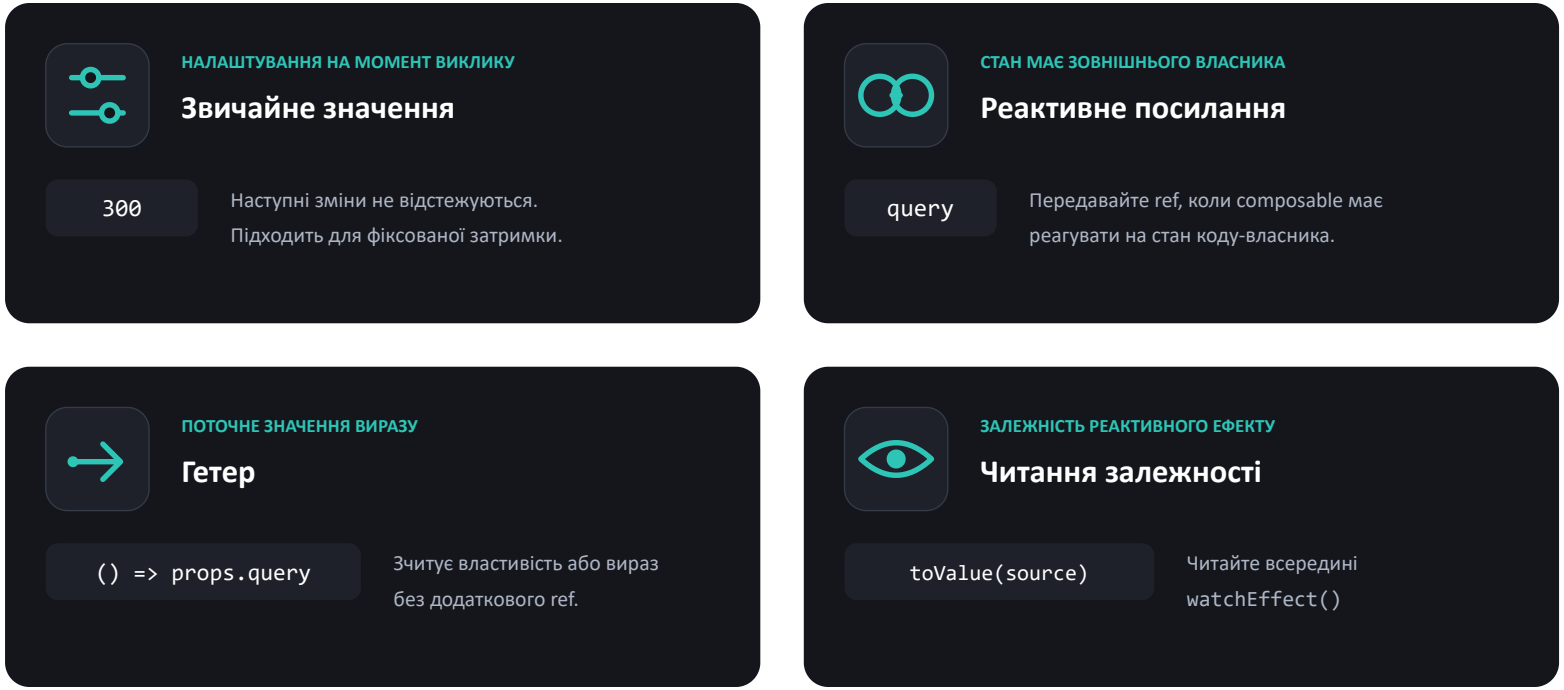


> `toValue()` нормалізує вхід усередині ефекту; окремі `ref` не втрачають реактивність після деструктуризації.

Vue.js — Input Arguments

Проєктування composables у Vue

Форма параметра визначає реактивність

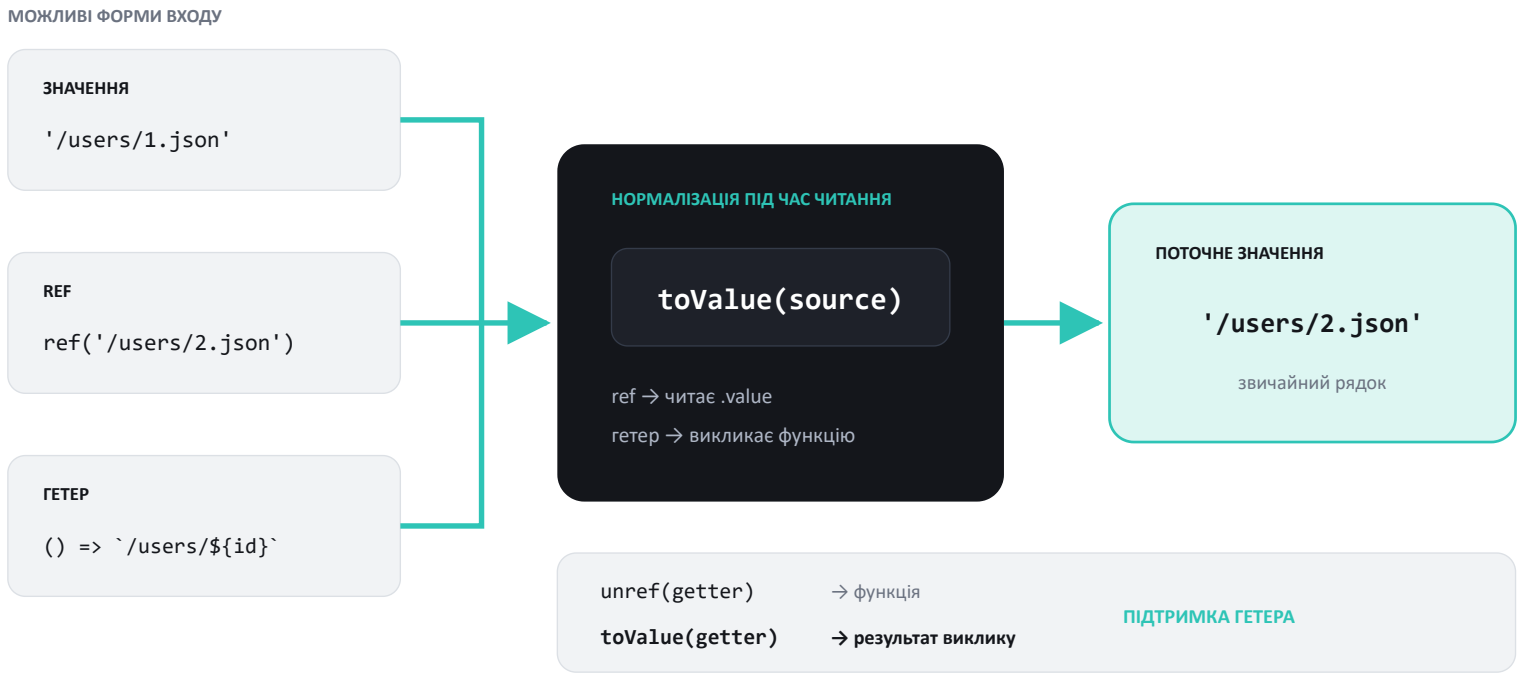


> Вибір залежить від того, чи має composable бачити наступні зміни джерела.

Vue.js — Input Arguments

Проєктування composables у Vue

toValue() зводить три форми входу до значення



> `toValue()` читає значення зараз; повторне читання організовує реактивний ефект.

Vue.js — toValue

Проєктування composables у Vue

Одна функція читає значення, ref і гетер

src/toValueInputs.js

```
import { ref, toValue } from 'vue'

const literal = 10
const reference = ref(20)
const getter = () => 30

export const normalized = [
  toValue(literal),
  toValue(reference),
  toValue(getter),
]
```

src/unrefVsToValue.js

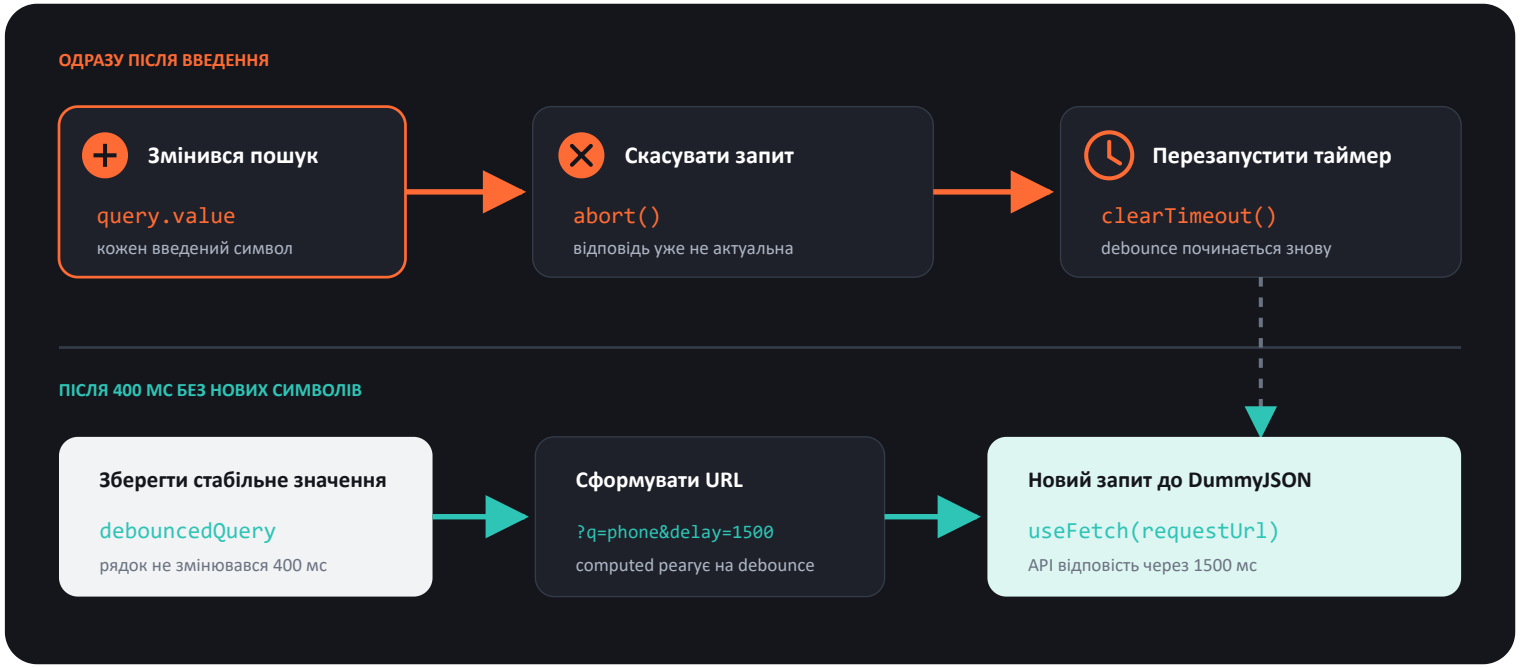
```
import { toValue, unref } from 'vue'

const getter = () => 30

export const fromUnref = unref(getter)
export const fromToValue = toValue(getter)
```

```
npm install && npm run dev
```

Зміна пошуку скасовує запит раніше за debounce



> `query` керує негайним `abort()`, а `debouncedQuery` — запуском наступного запиту.

DummyJSON — Search products

Сирий query скасовує, відкладений query запускає

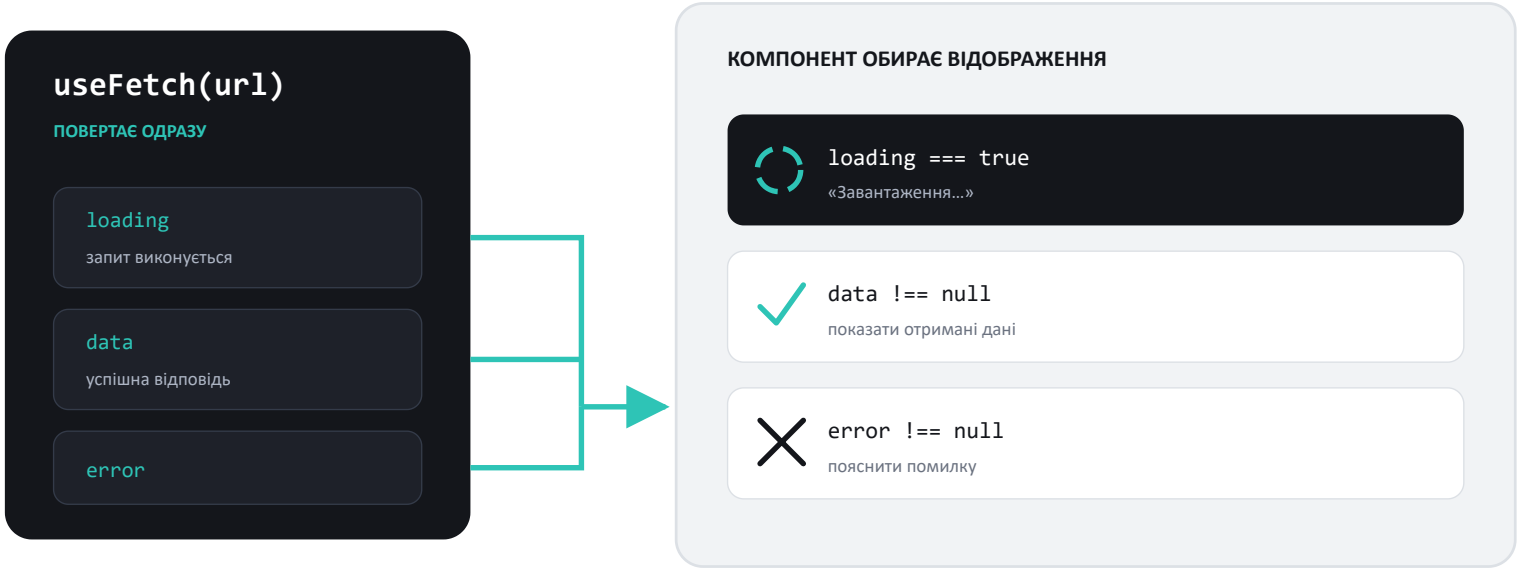
src/App.vue · рядки 14–22

```
const query = ref('')
const skip = ref(0)
const debouncedQuery = useDebouncedValue(query, 400)
const requestUrl = computed(() =>
  createProductsUrl(debouncedQuery.value, skip.value),
)
const { data, error, isLoading, abort, refetch } = useFetch(requestUrl)

watch(query, () => abort(), { flush: 'sync' })
```

```
npm install
```

useFetch повертає стан раніше за відповідь



Promise завершується пізніше; шаблон реагує на зміни ref автоматично.

> Компонент одразу отримує три `ref` і перемикає відображення після їхніх змін.

Vue.js — Async State Example

Проєктування composables у Vue

Компонент передає URL і відображає стан

src/FetchUsers.vue · рядки 1–12

```vue
<script setup>
import { ref } from 'vue'
import { useFetch } from './composables/useFetch.js'

const id = ref(1)
const fixed = useFetch('/users/1.json')
const url = ref('/users/2.json')
const reactive = useFetch(url)
const derived = useFetch(
  () => `/users/${id.value}.json`,
)
</script>
```

src/FetchUsers.vue · рядки 14–26

```vue
<template>
  <section>
    <h2>Користувач</h2>
    <button @click="id = 1">№ 1</button>
    <button @click="id = 2">№ 2</button>
    <button @click="id = 3">№ 3</button>
    <p v-if="derived.loading">Завантаження…</p>
    <p v-else-if="derived.error">
      {{ derived.error.message }}
    </p>
    <pre v-else>{{ derived.data }}</pre>
  </section>
</template>
```

```
npm install && npm run dev
```

Вхід може змінюватися після виклику

src/composables/useDebouncedValue.js

```
import { ref, toValue, watchEffect } from 'vue'

export function useDebouncedValue(source, delay = 300) {
  const value = ref(toValue(source))

  watchEffect((onCleanup) => {
    const next = toValue(source)
    const timer = setTimeout(() => {
      value.value = next
    }, delay)
    onCleanup(() => clearTimeout(timer))
  })

  return { value }
}
```

src/DebouncedSearch.vue

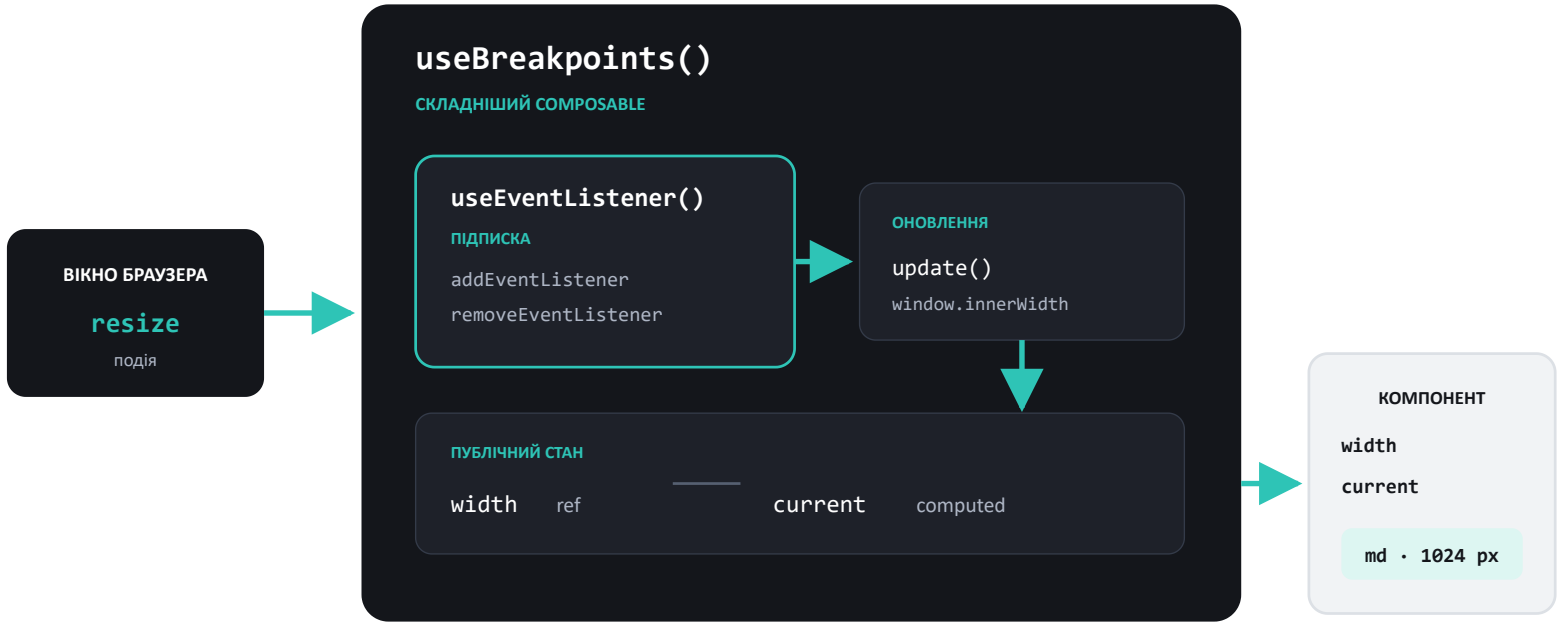
```
<script setup>
import { ref } from 'vue'
import {
  useDebouncedValue,
} from './composables/useDebouncedValue.js'

const query = ref('')
const { value: delayedQuery } = useDebouncedValue(query)
</script>

<template>
  <label>
    Пошук <input v-model="query" />
  </label>
  <p>Відкладене значення: {{ delayedQuery }}</p>
</template>
```

```
npm install && npm run dev
```

useBreakpoints складається з менших механізмів



> `useEventListener()` володіє підпискою; `useBreakpoints()` — шириною та поточним діапазоном.

Vue.js — Nested Composables

Проєктування composables у Vue

useBreakpoints + useEventListener

src/composables/useEventListener.js

```
import { onMounted, onUnmounted, toValue } from 'vue'

export function useEventListener(
  target, event, callback,
) {
  let element

  onMounted(() => {
    element = toValue(target)
    element?.addEventListener(event, callback)
  })

  onUnmounted(() => {
    element?.removeEventListener(event, callback)
  })
}
```

src/composables/useBreakpoints.js

```
import { computed, onMounted, ref } from 'vue'
import { useEventListener } from './useEventListener.js'
export function useBreakpoints(options = {}) {
  const { sm = 640, md = 768 } = options
  const width = ref(0)
  const current = computed(() =>
    width.value < sm ? 'base'
      : width.value < md ? 'sm'
        : 'md',
  )
  function update() {
    width.value = window.innerWidth
  }
  onMounted(update)
  useEventListener(() => window, 'resize', update)

  return { width, current }
}
```

```
npm install && npm run dev
```

Місце створення ref визначає власника



> `ref` усередині функції належить окремому виклику; `ref` на рівні модуля є спільним.

Vue.js — State Management

Проєктування composables у Vue

Два виклики отримують окремий або спільний стан

src/composables/useCounter.js

```
import { readonly, ref } from 'vue'

export function useLocalCounter() {
  const count = ref(0)
  return {
    count: readonly(count),
    increment: () => count.value++,
  }
}

const sharedCount = ref(0)

export function useSharedCounter() {
  return {
    count: readonly(sharedCount),
    increment: () => sharedCount.value++,
  }
}
```

src/CounterOwnership.vue · рядки 1–11

```
<script setup>
import {
  useLocalCounter,
  useSharedCounter,
} from './composables/useCounter.js'

const localA = useLocalCounter()
const localB = useLocalCounter()
const sharedA = useSharedCounter()
const sharedB = useSharedCounter()
</script>
```

```
npm install && npm run dev
```

Наступний крок

Перевірити межі браузера та SSR