Спеціальні директиви

Custom Directives

https://ua.vuejs.org/guide/reusability/custom-directives.html

Спеціальні директиви призначені для повторного використання логіки, яка передбачає низькорівневий доступ DOM до простих елементів

Загальна форма:

```
<тег  v-директива : аргумент . модифіктор = “ значення ” />
```

Приклад:

```
<div v-example:foo.bar="baz">
```

```
<script setup>
  const vExample = {
    mounted: (el, binding) => {
          . . . . . .
    }
  }
</script>
```

```
{
  arg: 'foo',
  modifiers: { bar: true },
  value: /* значення `baz` */,
  oldValue: /* значення `baz` з попереднього оновлення */
}
```

Спеціальні директиви призначені для повторного використання логіки, яка передбачає низькорівневий доступ DOM до простих елементів

Загальна форма:

```
<тег    v-директива : аргумент . модифіктор = " значення " />
```

```js
const myDirective = {
  // викликається перед прив'язкою атрибутів елемента
  // або застосованням слухачів подій
  created(el, binding, vnode, prevVnode) {
    // подробиці аргументів див. нижче
  },
  // викликається безпосередньо перед тим, як елемент буде вставлено в DOM.
  beforeMount(el, binding, vnode, prevVnode) {},
  // викликається, коли зв'язаний елемент є батьківським компонентом
  // і всі його діти змонтовані.
  mounted(el, binding, vnode, prevVnode) {},
  // викликається перед оновленням батьківського компонента
  beforeUpdate(el, binding, vnode, prevVnode) {},
  // викликається після оновлення
  // батьківського компонента й усіх дочірніх елементів
  updated(el, binding, vnode, prevVnode) {},
  // викликається перед тим, як батьківський компонент буде демонтовано
  beforeUnmount(el, binding, vnode, prevVnode) {},
  // викликається, коли батьківський компонент демонтовано
  unmounted(el, binding, vnode, prevVnode) {}
}
```

```
----- binding ------
{
  arg: 'аргумент',
  modifiers: { модифікатор: true },
  value: /* значення `baz` */,
  oldValue: /* поп.значення `baz`*/
}
```

Спеціальні директиви призначені для повторного використання логіки, яка передбачає низькорівневий доступ DOM до простих елементів

Загальна форма: <тег v-директива : аргумент . модифіктор = “ значення ” />

Приклад: Обрізати текст до 10 символів

```
<template>
  <div>
    <h3>Використано 10 символів</h3>
    <div v-ellipses:10>Hello my dear friends! Welcome to my side.</div>
```

Використано 10 символів

Hello my d...

Спеціальні директиви призначені для повторного використання логіки, яка передбачає низькорівневий доступ DOM до простих елементів

Загальна форма: <тег v-директива : аргумент . модифіктор = “ значення ” />

Приклад:
Обрізати текст до 10 символів

```
<div v-ellipses:10>
```

```
const vEllipses = {

}
```

Створюємо об’єкт з назвою vДирктива => vEllipses

Спеціальні директиви призначені для повторного використання логіки, яка передбачає низькорівневий доступ DOM до простих елементів

Загальна форма: <тег v-директива : аргумент . модифіктор = “ значення ” />

Приклад:
Обрізати текст
до 10 символів

```
<div v-ellipses:10>
```

```
const vEllipses = {
 mounted: (el, binding) => {
```

```
binding = {
            arg: ‘10',
          }
```

Додаємо хук обробки (created, mounted, updated, …)

```
}
}
```

Спеціальні директиви призначені для повторного використання логіки, яка передбачає низькорівневий доступ DOM до простих елементів

Загальна форма: <тег v-директива : аргумент . модифіктор = “ значення ” />

Приклад:
Обрізати текст
до 10 символів

```
<div v-ellipses:10>
```

```
const vEllipses = {
  mounted: (el, binding) => {
    const contentStr = el.innerText

}
}
```

Отримуємо вміст цільового елемента el

```
binding = {
            arg: ‘10',
          }
```

Спеціальні директиви призначені для повторного використання логіки, яка передбачає низькорівневий доступ DOM до простих елементів

Загальна форма: <тег v-директива : аргумент . модифіктор = “ значення ” />

Приклад:
Обрізати текст
до 10 символів

```
<div v-ellipses:10>
```

```
const vEllipses = {
  mounted: (el, binding) => {
    const contentStr = el.innerText

    let charsNumber = parseInt(binding.arg)

}
}
```

Отримуємо значення аргументу: arg

```
binding = {
            arg: ‘10',
          }
```

Спеціальні директиви призначені для повторного використання логіки, яка передбачає низькорівневий доступ DOM до простих елементів

Загальна форма: <тег v-директива : аргумент . модифіктор = “ значення ” />

Приклад:
Обрізати текст
до 10 символів

```
<div v-ellipses:10>
```

```
const vEllipses = {
  mounted: (el, binding) => {
    const contentStr = el.innerText

    let charsNumber = parseInt(binding.arg)

    if (contentStr.length > charsNumber)
        el.innerText = contentStr.substring(0, charsNumber) + '...'
  }
}
```

```
binding = {
            arg: ‘10',
          }
```

Виконуємо необхідні операції

Спеціальні директиви призначені для повторного використання логіки, яка передбачає низькорівневий доступ DOM до простих елементів

Загальна форма: <тег v-директива : аргумент . модифіктор = “ значення ” />

Приклад: <div v-ellipses:10>

```
<script setup>
const vEllipses = {
  mounted: (el, binding) => {
    if (el.innerText.length > binding.arg)
      el.innerText = el.innerText.substring(0, binding.arg) + '...'
  }
}
</script>
```

```
<template>
  <div>
    <h3>Використано 10 символів</h3>
    <div v-ellipses:10>Hello my dear friends! Welcome to my side.</div>

    <h3>Використано 15 символів</h3>
    <div v-ellipses:15>Hello my dear friends! Welcome to my side.</div>
  </div>
</template>
```

Використано 10 символів
Hello my d...
Використано 15 символів
Hello my dear f...

Спеціальні директиви призначені для повторного використання логіки, яка передбачає низькорівневий доступ DOM до простих елементів

Загальна форма: <тег v-директива : аргумент . модифіктор = “ значення ” />

Приклад: <div v-ellipses:10.keep-word>

Обрізати текст до 10 символів, але не розривати слова

```
const vEllipses = {
  mounted: (el, binding) => {
    const contentStr = el.innerText
    let charsNumber = parseInt(binding.arg)
    if (el.innerText.length > charsNumber)
      if (binding.modifiers['keep-word']) {      const spaceOrPunctuation = /[\s\.,;:!?'"]/
            const position = contentStr.slice(charsNumber).search(spaceOrPunctuation)
            charsNumber = charsNumber + position
      }
    el.innerText = el.innerText.substring(0, charsNumber) + '...'
  }
}
```

Спеціальні директиви призначені для повторного використання логіки, яка передбачає низькорівневий доступ DOM до простих елементів

Загальна форма: `<тег v-директива : аргумент . модифіктор = “ значення ” />`

Приклад: `<div v-ellipses:10.keep-word>`

```
<script setup>
const vEllipses = {
  mounted: (el, binding) => {
    const contentStr = el.innerText
    let charsNumber = parseInt(binding.arg)
    if (el.innerText.length > charsNumber) {
      if (binding.modifiers['keep-word']) {
        const spaceOrPunctuation = /[\s\.,;:!?'"]/
        const position = contentStr.slice(charsNumber).search(spaceOrPunctuation)
        charsNumber = charsNumber + position
      }

      el.innerText = el.innerText.substring(0, charsNumber) + '...'
    }
  }
}
</script>
```

```
<h3>Використано 10 символів</h3>
<div v-ellipses:10>Hello my dear friends! Welcome to my site.</div>

<h3>Використано 10 символів (слова не перериваємо)</h3>
<div v-ellipses:10.keep-word>Hello my dear friends! Welcome to my site.</div>
```

Використано 10 символів

Hello my d...

Використано 10 символів (слова не перериваємо)

Hello my dear...

Спеціальні директиви призначені для повторного використання логіки, яка передбачає низькорівневий доступ DOM до простих елементів

Загальна форма: `<тег v-директива : аргумент . модифіктор = “ значення ” />`

Приклад:

Обрізати текст, не розривати слова,вибирати кінцевий символ

```
<div v-ellipses:10.keep-word=“ ‘ ___ ’ ">
```

```
const vEllipses = {
  mounted: (el, binding) => {
    const contentStr = el.innerText
    let charsNumber = parseInt(binding.arg)
    if (el.innerText.length > charsNumber)
      if (binding.modifiers['keep-word']) {       const spaceOrPunctuation = /[\s\.,;:!?'"]/
            const position = contentStr.slice(charsNumber).search(spaceOrPunctuation)
            charsNumber = charsNumber + position
      }
    el.innerText = el.innerText.substring(0, charsNumber) + (binding.value ?? '...‘)
  }
}
```

Спеціальні директиви призначені для повторного використання логіки, яка передбачає низькорівневий доступ DOM до простих елементів

Загальна форма: <тег v-директива : аргумент . модифіктор = “ значення ” />

Приклад: <div v-ellipses:10.keep-word=“ ‘___’ ">

```
<script setup>
const vEllipses = {
  mounted: (el, binding) => {
    const contentStr = el.innerText
    let charsNumber = parseInt(binding.arg)
    if (el.innerText.length > charsNumber) {
      if (binding.modifiers['keep-word']) {
        const spaceOrPunctuation = /[\s\.,;:!?'"]/
        const position = contentStr.slice(charsNumber).search(spaceOrPunctuation)
        charsNumber = charsNumber + position
      }

      el.innerText = el.innerText.substring(0, charsNumber) + (binding.value ?? '...')
    }
  }
}
</script>
<h3>Використано 15 символів</h3>
<div v-ellipses:15>Hello my dear friends! Welcome to my site.</div>

<h3>Використано 10 символів (слова не перериваємо)</h3>
<div v-ellipses:10.keep-word>Hello my dear friends! Welcome to my site.</div>

<h3>Використано 10 символів</h3>
<div v-ellipses:10.keep-word="'___'">Hello my dear friends! Welcome to my site.</div>
```

```
Використано 15 символів
Hello my dear f...
Використано 10 символів (слова не перериваємо)
Hello my dear...
Використано 10 символів
Hello my dear___
```