ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
Сучасні каркаси web-додатків
Лісняк Андрій Олександрович

Лабораторна робота 3. Створення адаптивного модального вікна з використанням слотів, ref, Teleport і Transition

Мета

Навчитися створювати перевикористовуваний адаптивний компонент модального вікна, передавати його вміст слотами та керувати станом, розміщенням, переходами й фокусом засобами Vue.

Теоретичні відомості

Пропси забезпечують передачу структурованих значень, проте вони не призначені для передавання довільних фрагментів розмітки з власними полями чи кнопками. Для гнучкого компонування компонентів використовують механізм слотів.

Слот (slot) — зарезервоване місце у шаблоні компонента, призначене для виведення розмітки, яку передає батьківський компонент.

Точка виведення слота — тег `<slot>`, розміщений у шаблоні дочірнього компонента, на місце якого підставляється переданий вміст.

Запасний вміст (fallback content) — розмітка між тегами `<slot>` і `</slot>`, яка відображається за відсутності переданого вмісту від батьківського компонента.

Елемент `<slot>` позначає межу, всередині якої батьківський компонент керує представленням. Дочірній компонент надає оболонку, але не обмежує тип елементів, розташованих усередині. Якщо компонент містить один елемент `<slot />` без атрибута `name`, він приймає весь переданий вміст як основний слот за замовчуванням.

Іменований слот — слот із явно заданим атрибутом `name`, який дозволяє передавати декілька незалежних фрагментів розмітки до визначених областей одного компонента.

Наприклад, для розділення областей інтерфейсу компонент діалогу оголошує неіменований `<slot />` для форми або повідомлення та `<slot name="footer" />` для кнопок дій. Батьківський компонент передає вміст для іменованого слота через тег `<template #footer>`, де символ `#` є скороченням директиви `v-slot:`.

Область видимості слота — контекст даних, у якому обчислюються змінні та вирази переданої розмітки.

Вміст, переданий у слот, компілюється в контексті батьківського компонента й має доступ до його реактивного стану. Так, наприклад, кнопка збереження у слоті `#footer` викликає метод батьківського каталогу безпосередньо, а дочірній діалог відповідає лише за її розміщення.

Слот із параметрами (scoped slot) — слот, який передає локальні дані дочірнього компонента у батьківський шаблон через пропси слота.

Пропси слота (slot props) — атрибути, зв'язані з елементом `<slot>` у дочірньому компоненті, які стають доступними у значенні директиви `v-slot` батьківського шаблону.

За замовчуванням, вміст слота не має доступу до внутрішніх змінних дочірнього компонента. Якщо батьківському шаблону потрібні такі дані (наприклад, внутрішній стан обробки чи допоміжна функція), дочірній компонент передає їх як атрибути елемента `<slot>`.

```
<!-- Дочірній компонент ModalDialog.vue -->
<template>
  <div class="dialog">
    <slot/>
    <footer class="dialog-footer">
      <slot name="footer" :close="close" :is-
submitting="isSubmitting"/>
    </footer>
  </div>
</template>
```

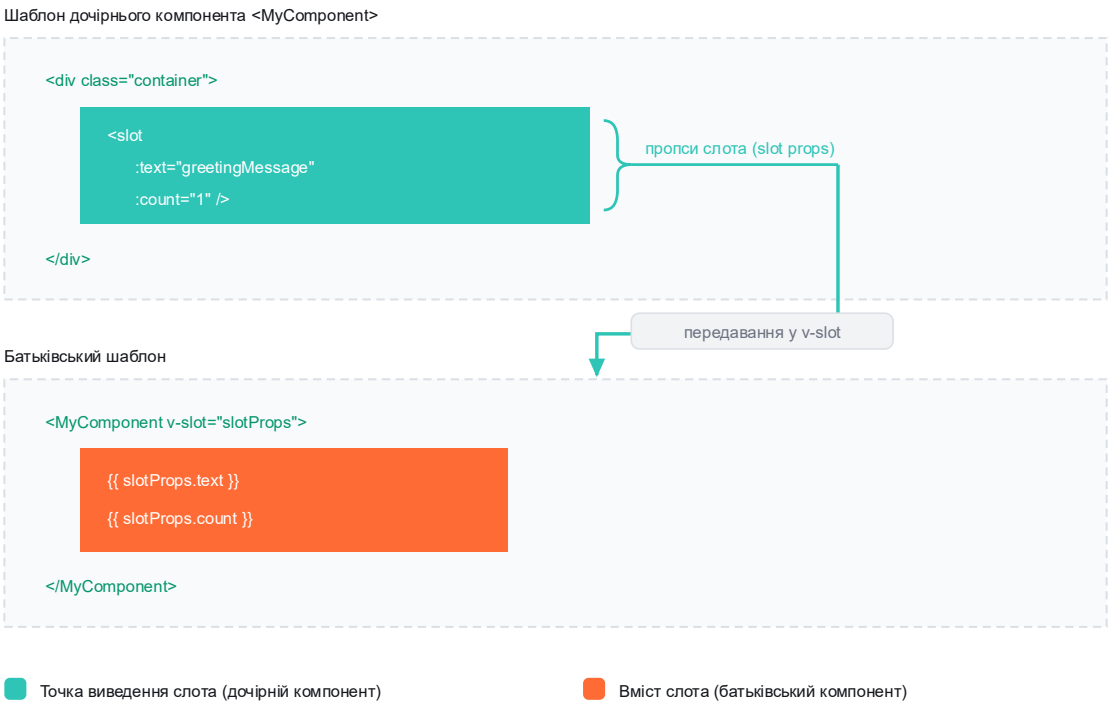

Рисунок 1 — Передавання даних у батьківський шаблон через пропси слота.

Батьківський компонент отримує об'єкт переданих параметрів через директиву `v-slot` або деструктуризацію після символу `#`.

```
<!-- Батьківський компонент -->
<template>
  <ModalDialog v-model="isOpen">
    <p>Редагування запису</p>

    <template #footer="{ close, isSubmitting }">
      <button type="button" @click="close">Скасувати</button>
      <button type="button" :disabled="isSubmitting"
@click="save">Зберегти</button>
    </template>
  </ModalDialog>
</template>
```

Такий механізм поєднує інкапсуляцію поведінки в дочірньому компоненті з можливістю для батьківського компонента самостійно визначати розмітку елементів керування.

Реактивні значення та посилання на DOM

Реактивне посилання ref — об’єкт Vue, який зберігає значення у властивості `value` та дозволяє відстежувати його зміни в системі реактивності.

Шаблонне посилання (template ref) — механізм прямого доступу до змонтованого DOM-елемента або екземпляра дочірнього компонента.

Функція `ref(false)` створює реактивний логічний стан, тоді як `ref(null)` ініціалізує змінну для збереження майбутнього посилання на вузол DOM. У JavaScript значення читають і змінюють через властивість `.value`, а в шаблоні верхньорівневі реактивні змінні розгортаються автоматично.

Прямий доступ до DOM-елемента необхідний для керування фокусом (`el.focus()`), зчитування геометричних розмірів або взаємодії з браузерними API, де декларативних директив недостатньо.

Статичні та динамічні шаблонні посилання

Шаблонне посилання зв'язує змінну з елементом через спеціальний атрибут `ref`.

У базовому варіанті значення атрибута є статичним рядком, який збігається з назвою змінної, створеної у `<script setup>`.

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

const inputElement = ref(null)

onMounted(() => {
  inputElement.value?.focus()
})
</script>

<template>
  <input ref="inputElement" type="text" />
</template>
```

Для складніших сценаріїв застосовують динамічне зв'язування за допомогою директиви `v-bind:ref` або скороченого запису `:ref`.

Функція зворотного виклику рефу (function ref) — функція, передана в атрибут `:ref`, яка викликається під час кожного оновлення компонента та отримує цільовий DOM-елемент як аргумент.

```
<template>
  <input :ref="(el) => { currentInput = el }" type="text" />
</template>
```

Функція зворотного виклику отримує посилання на DOM-елемент під час монтування або оновлення, а під час демонтажу елемента — значення `null`. Демонтажем є вилучення елемента з дерева документа, зокрема директивою `v-if` або видаленням запису зі списку. Приховування засобами CSS елемент не демонтує, тому після `v-show` чи властивості `display: none` посилання залишається чинним і вказує на той самий вузол. Такий підхід дозволяє динамічно зберігати посилання на елементи списку у масив або мапу без створення фіксованої кількості окремих змінних.

Одержане шаблонне посилання передають далі як звичайне значення пропса. У прикладі нижче `initial-focus` і `fallback-focus` — оголошені пропси компонента ModalDialog, а префікс `:` є скороченням директиви `v-bind`, а не атрибутом `ref`. Батьківський компонент зберігає посилання на елементи у власних змінних `nameInput` і `catalogHeading`, а дочірній одержує вже готові DOM-елементи й викликає на них `focus()`.

```
<!-- Передавання шаблонного посилання як пропсу для керування
фокусом -->
<ModalDialog
  v-model="isOpen"
  :initial-focus="nameInput"
  :fallback-focus="catalogHeading"
/>
```

Життєвий цикл і часові межі доступу до DOM

Цикл оновлення DOM — асинхронний процес, у якому Vue накопичує зміни реактивного стану й застосовує їх до дерева документа однією партією після завершення поточної синхронної роботи.

До монтування компонента та після видалення елемента з дерева за допомогою директиви `v-if` шаблонне посилання завжди дорівнює `null`. Зміна реактивної змінної не оновлює структуру DOM синхронно в тому самому рядку коду.

Для очікування завершення оновлення DOM використовують виклик `await nextTick()`. Він завершується після того, як Vue застосувало накопичені зміни до дерева документа. Перерахунок геометрії та малювання виконує

браузер окремим етапом, тому `nextTick` не гарантує ні їх завершення, ні завершення анімацій та переходів CSS. Для встановлення фокуса всередині анімованого елемента використовують подію `after-enter` компонента `Transition`, а для очищення ресурсів — подію `after-leave`.

Розміщення через Teleport

DOM — об'єктне дерево документа, з яким браузер пов'язує елементи сторінки та їхню взаємодію.

Teleport — вбудований компонент Vue, який виводить частину шаблону в іншому DOM-контейнері, зберігаючи логічну ієрархію компонентів.

Панель, вкладена у картку, може потрапити під обмеження контексту накладання її предків. Властивість `transform` предка також змінює поведінку позиціювання `fixed`. Саме лише велике `z-index` не усуває ці обмеження. `<Teleport to="body">` переносить розмітку за межі контейнера застосунку. Пропси, події та контекст слотів при цьому не змінюються. Контейнер призначення має існувати до монтування Teleport.

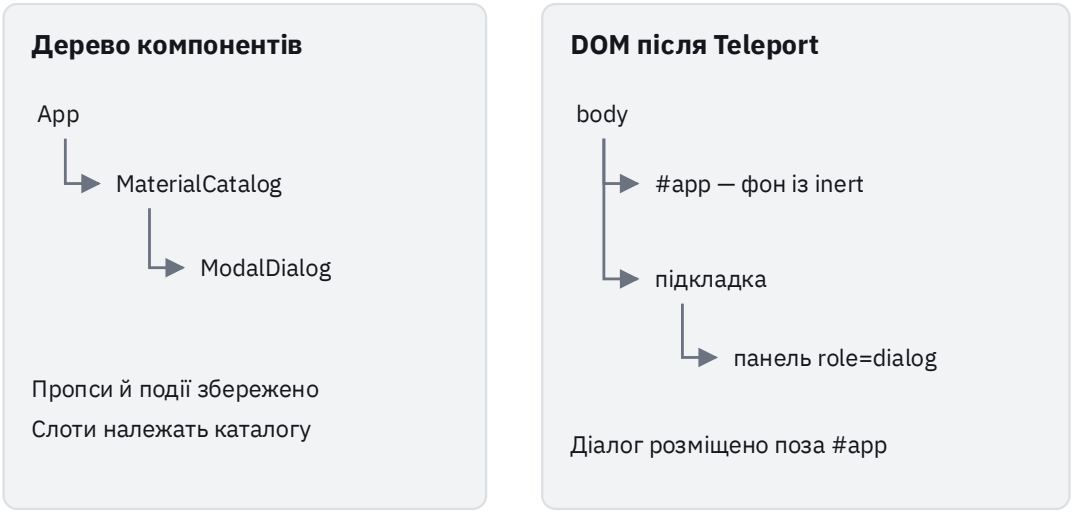

Рисунок 2 — Логічний зв'язок зберігається після перенесення DOM. Схема автора курсу.

Teleport сам по собі не створює модальність, не блокує фон і не керує фокусом. Ці властивості забезпечує компонент діалогу.

Переходи появи та зникнення

Transition — вбудований компонент Vue, який узгоджує вставлення, видалення або зміну видимості дочірнього елемента з анімацією.

Transition має один кореневий дочірній елемент. Під час появи `enter-from-class` задає початковий стан, `enter-active-class` — властивості переходу, `enter-to-class` — кінцевий стан. Для зникнення служать відповідні параметри `leave-*`. У них можна передавати повні класи Tailwind, наприклад `opacity-0`, `opacity-100`, `transition-opacity` і `duration-150`.

Під час закриття елемент ще присутній у DOM, доки триває перехід. Якщо прибрати весь ModalDialog зовнішнім `v-if`, його внутрішня анімація може не завершитися. Тому компонент залишається змонтованим, а умова керує коренем усередині Transition. Розблокування фону виконують після `after-leave`, а також під час демонтажу як очищення ресурсів. Для користувачів із `prefers-reduced-motion: reduce` перехід вимикають; логіка закриття не повинна залежати від таймера фіксованої тривалості.

Адаптивне оформлення й доступність

Адаптивний інтерфейс — інтерфейс, компонування якого пристосовується до доступного простору зі збереженням змісту та можливості взаємодії.

Утилітарний клас — CSS-клас, що задає невелику окрему властивість оформлення, наприклад відступ або спосіб розташування елементів.

Tailwind застосовує базові класи на всіх ширинах, а варіанти `sm:`, `md:` та інші — починаючи з відповідного порога. У типовій конфігурації версії 3 поріг `sm` становить 640 px. Отже, `flex-col sm:flex-row` змінює вертикальне розташування кнопок на горизонтальне від цього порога. Класи мають бути присутні в коді повністю, щоб Tailwind виявив їх під час збірки.

Максимальна ширина обмежує довжину рядків на великому екрані. Обмеження висоти з урахуванням `100dvh` враховує поточну висоту області перегляду. У панелі з flex-компонуванням заголовок і нижня область не стискаються, а основний вміст має `min-h-0 overflow-y-auto`. Це дозволяє прокручувати довгий текст без втрати кнопок.

Фокус клавіатури — поточний елемент, який отримує клавіатурне введення.

Атрибут inert — ознака, яка вилучає елемент і його нащадків із взаємодії та дерева доступності браузера.

Відкрите модальне вікно отримує фокус і утримує послідовність Tab усередині себе. Під час підтвердження незворотної дії початковий фокус доцільно спрямувати на скасування. Після закриття фокус повертають до елемента виклику або логічного наступного місця, якщо його вже немає.

Панель позначають `role="dialog"`, `aria-modal="true"` і пов’язують із заголовком через `aria-labelledby`. Ці атрибути описують семантику, але не реалізують блокування. Для фону застосовують `inert` і тимчасово блокують прокручування. Початкові значення властивостей зберігають, щоб після закриття відновити попередній стан сторінки.

Функціональні вимоги

- Один компонент ModalDialog використано для редагування матеріалу та підтвердження його видалення; одночасно відкрите не більше одного вікна

- Форма редагує назву й тривалість; назва після trim непорожня, а тривалість є додатним цілим числом
- Каталог змінюється лише після успішного збереження або підтвердження видалення; будь-який спосіб скасування відкидає чернетку
- Після зміни чи видалення узгоджено оновлюються список, фільтрація, кількість, сумарна тривалість і вибір за id; для порожнього результату показано повідомлення
- Вікно закривається хрестиком, клавішею Escape, кнопкою Скасувати та натисканням безпосередньо на підкладку; натискання всередині панелі не закриває його
- Початковий фокус потрапляє в поле назви під час редагування та на кнопку Скасувати під час підтвердження видалення; Tab і Shift+Tab не виходять за межі вікна
- Після закриття фокус повертається на кнопку виклику, а якщо вона зникла або прихована, на заголовок каталогу
- Фон не реагує на введення й не прокручується до завершення анімації закриття; після закриття та демонтажу початкові властивості фону відновлюються
- За розмірів 360×800, 768×1024, 1440×900 і 844×390 немає горизонтальної прокрутки, обрізаного тексту чи недоступних елементів керування
- Ширина панелі не перевищує 640 px, зовнішні відступи становлять щонайменше 16 px, довгий вміст прокручується всередині, а заголовок і кнопки лишаються доступними
- Кнопки footer розташовано вертикально на ширині меншій за 640 px та горизонтально від 640 px; області натискання кнопок і висота полів становлять щонайменше 44 px
- Поява та зникнення плавні; за prefers-reduced-motion рух і тривалі переходи вимкнено, а керування фокусом працює
- Сторінка та обидва вікна містять видимі ПІБ, групу й відомості про середовище для ідентифікації знімків екрана

Технічні вимоги

- Vue 3 SFC, JavaScript, Composition API та script setup; готові бібліотеки діалогів і анімації не використовуються
- ModalDialog має пропси modelValue і title, подію update:modelValue, слот за замовчуванням та іменований слот footer; предметні дії залишаються у MaterialCatalog
- Стан відкриття, список і чернетка створені через ref(); шаблонні ref використовуються для панелі, початкового фокуса й заголовка каталогу
- Єдиний ModalDialog постійно змонтований, Teleport спрямовано в body, а v-if розміщено на єдиному кореневому елементі всередині Transition

- Оформлення діалогу й нових елементів виконано повними класами Tailwind CSS 3.4.17; класи переходу передано параметрами Transition; CSS попереднього каталогу дозволено
- Tailwind підключено локально через PostCSS з охопленням vue-файлів у content; версії встановлених залежностей відтворюються за package-lock.json
- Панель має role=dialog, aria-modal=true і aria-labelledby на унікальний заголовок; поля мають label, кнопки доступні назви та видимий focus-visible
- Фоновий контейнер app отримує inert, попередній overflow body зберігається; відновлення виконується після after-leave та під час демонтажу, без довільних таймерів
- Ідентифікатор матеріалу є стабільним ключем і критерієм вибору; пропси та імпортований початковий масив не змінюються безпосередньо
- Дані зберігаються в пам’яті; HTTP-запити, маршрутизація та localStorage не використовуються

Необхідне обладнання та матеріали

- Власний Vue-проєкт лабораторної роботи 2 та його package-lock.json
- Node.js 24 LTS, npm, Git і редактор коду
- Tailwind CSS 3.4.17, PostCSS 8, Autoprefixer 10
- Браузер із DevTools і режимом емуляції розмірів екрана

Хід роботи

Розширте власний каталог навчальних матеріалів універсальним компонентом ModalDialog. Покажіть два його застосування — редагування назви й тривалості та підтвердження видалення.

1. Підготовка проєкту

У корені власного репозиторію створіть копію проєкту попередньої роботи.

2. Підключення Tailwind

У каталозі нової роботи встановіть локальні залежності:

```
npm install -D --save-exact tailwindcss@3.4.17 postcss@8.5.6
autoprefixer@10.4.21
npx tailwindcss init -p
```

Для ESM-проєкту налаштуйте `tailwind.config.js`:

```
export default {
    content: ['./index.html', './src/**/*.{vue,js}'],
    corePlugins: {preflight: false},
    theme: {extend: {}},
    plugins: [],
}
```

Preflight вимкнено, щоб глобальне скидання стилів не змінило надану BEM-розмітку каталогу. Наприкінці `src/style.css` додайте:

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Переконайтеся, що `main.js` імпортує цей файл. Тимчасово додайте `p-4` до нового елемента й перевірте відступ у DevTools. Збережіть оновлений lock-файл. Власний CSS каталогу можна залишити; діалог, нові кнопки та поля оформлюйте класами Tailwind.

3. Контракт і слоти ModalDialog

Створіть `src/components/ModalDialog.vue`. Він приймає булевий `modelValue` зі значенням за замовчуванням `false` та обов'язковий рядок `title`, оголошує `update:modelValue` і не змінює пропси.

Усередині панелі передбачте заголовок, кнопку закриття, слот за замовчуванням для основного вмісту та іменований слот `footer` для кнопок. Заголовок одержує унікальний `id`, а панель — `role="dialog"`, `aria-modal="true"` та `aria-labelledby` на цей заголовок. Для одного постійного екземпляра припустимий сталий унікальний id.

У MaterialCatalog створіть `isOpen = ref(false)` та `dialogTitle` з початковим текстом заголовка, після чого підключіть компонент:

```
<ModalDialog v-model="isOpen" :title="dialogTitle">
  <!-- Форма або текст підтвердження -->
  <template #footer>
    <!-- Кнопки поточного сценарію -->
  </template>
</ModalDialog>
```

Це демонструє різницю між пропсом і слотом. Заголовок є значенням, а форма й кнопки — розміткою з обробниками батьківського компонента. Перевірте, що зміна тексту та кнопок не потребує редагування ModalDialog.

4. Teleport і структура розмітки — 10 хвилин

У ModalDialog використайте вкладеність `Teleport → Transition → підкладка з v-if → панель`. Teleport спрямовується в `body`, а умова залежить від `modelValue`. Не ставте `v-if` на сам ModalDialog у каталозі.

Підкладка має перекривати область перегляду через `fixed inset-0`. Панель залишається її дочірнім елементом.

На цьому етапі `isOpen` дорівнює `false`, тому підкладки в DOM ще немає, а кнопки сценаріїв з'являться лише на етапі 5. Щоб відкрити вікно зараз, тимчасово додайте до розмітки каталогу кнопку:

```
<button type="button" @click="isOpen = true">Відкрити
(тимчасово)</button>
```

Відкрийте вікно цією кнопкою. Через DevTools переконайтеся, що підкладка міститься в `body` поза `#app`, хоча у Vue DevTools ModalDialog лишається нащадком каталогу. Збережіть знімок DOM для звіту.

5. Редагування матеріалу

Створіть реактивний робочий список копіюванням імпортованих об'єктів:

```
const items = ref(materials.map(material => ({...material})))
const selectedId = ref(items.value[0]?.id ?? null)
const draft = ref({title: '', minutes: ''})
```

Переведіть фільтрацію й підсумки на `items`. Зберігайте вибір за `selectedId`, а назву отримуйте з поточного об'єкта. Подія `select` картки може залишитися без змін; у батьківському обробнику використайте id поточного елемента циклу.

Додайте до картки кнопки `Редагувати` та `Видалити`. Їхні події повідомляють каталогові про вибраний матеріал. Тимчасову кнопку відкриття з етапу 4 приберіть. Перед відкриттям створіть свіжу чернетку з його назви й тривалості та збережіть id редагованого запису.

У слоті діалогу розмістіть форму з пов'язаними `label`, текстовим полем назви й числовим полем тривалості. Зв'яжіть поля з чернеткою через `v-model`. Оголосіть шаблонний ref поля назви для початкового фокуса.

Формі задайте унікальний `id="material-edit-form"` і `@submit.prevent`. Кнопці `Зберегти` у footer задайте `type="submit" form="material-edit-form"`. Так вона надсилає форму, хоча розміщена за її межами; Enter у полі працює так само.

Перевіряйте назву після `trim()`, а числове значення через `Number.isInteger()` та порівняння з нулем. Для некоректних даних покажіть зрозуміле повідомлення й залиште форму відкритою. Перевірку додайте до обробника, навіть якщо є HTML-атрибути `required`, `min` та `step`.

Після успіху замініть відповідний запис за id. Зміни назви можуть прибрати картку з відфільтрованого результату. Підсумки повинні перерахуватися через computed. До збереження каталог не змінюється.

6. Підтвердження видалення

Зберігайте режим `edit` або `delete` в каталозі й використовуйте той самий постійний екземпляр ModalDialog. Для видалення в основному слоті покажіть назву матеріалу та наслідок дії, у footer — кнопки `Скасувати` й `Видалити`. Початковий фокус спрямовується на скасування.

Видаляйте запис за id лише після підтвердження. Якщо видалено обраний матеріал, виберіть перший зі списку, що залишився, або `null`, якщо він порожній. Наявні фільтри зберігаються. Для порожнього списку покажіть повідомлення й нульові підсумки.

Не очищуйте режим і вміст одразу після перемикання `isOpen` у false. Вони мають залишатися видимими до завершення зникнення. Під час наступного відкриття заново заповніть чернетку й очистьте помилки попередньої форми.

7. Закриття, фокус і фон — 30 хвилин

Додайте всі способи скасування. Кнопка хрестика має доступну назву `Закрити вікно`; Escape обробляється всередині вікна. На підкладці використайте `@click.self`, щоб натискання на панель не закривало її.

Для узгодження фокуса дозвольте додаткові пропси `initialFocus` і `fallbackFocus` з DOM-елементами та значенням за замовчуванням `null`. Передавайте в них шаблонні ref поля назви або кнопки скасування та заголовка каталогу. Заголовок має `tabindex="-1"`. До появи елемента посилання може бути null.

У ModalDialog створіть шаблонний ref панелі та збережіть `document.activeElement` перед відкриттям. Після оновлення DOM переведіть фокус усередину; після `after-enter` уточніть його через `initialFocus`, якщо потрібно. Якщо ціль ще недоступна, фокусуйте панель із `tabindex="-1"`.

Під час відкриття збережіть попередні `inert` контейнера `#app` та `document.body.style.overflow`. Після перенесення фокуса в діалог зробіть фон inert і заблокуйте прокручування body. Не встановлюйте inert на body, адже воно містить і сам діалог.

Обробіть Tab та Shift+Tab на панелі. Визначайте актуальний перелік видимих доступних для фокуса елементів; виключайте приховані й disabled. Після останнього фокус переходить на перший, перед першим — на останній. Якщо доступних елементів немає, фокус залишається на панелі.

У `after-leave` відновіть початкові властивості фону. Якщо кнопка виклику ще приєднана до DOM і видима, поверніть фокус на неї. Інакше використайте `fallbackFocus`. Перед відновленням переконайтеся, що вікно не було повторно відкрито. Ту саму процедуру очищення викличте під час демонтажу; не додавайте довільний setTimeout.

8. Переходи й адаптивність

Передайте класи Tailwind параметрами Transition. Для єдиного кореня достатньо переходу прозорості зі стану `opacity-0` до `opacity-100`. Параметри `enter-active-class` та `leave-active-class` можуть містити `transition-`

`opacity duration-150 motion-reduce:transition-none`. Розблокування фону має працювати і без анімації.

Підкладці задайте `p-4 box-border flex items-center justify-center`. Панелі — `box-border w-full max-w-screen-sm max-h-[calc(100dvh-2rem)] flex flex-col overflow-hidden`. Ці класи записуються в одному атрибуті без розбиття самих назв. Явно задайте фон панелі й затемнення підкладки.

Заголовку та footer задайте `shrink-0`, основному вмісту — `min-h-0 overflow-y-auto break-words`. Для footer використайте `flex flex-col gap-3 sm:flex-row sm:justify-end`. Кнопки мають `min-h-11 min-w-11`, поля — `min-h-11`. Додайте видимий стан `focus-visible`; не прибирайте outline без заміни.

Нові поля мають вміщуватися у ширину панелі. У заголовку дозволено перенесення рядків, а хрестик не стискається.

9. Перевірка та завершення — 30 хвилин

Запустіть застосунок за адресою `http://localhost:5173`:

```
npm run dev -- --port 5173 --strictPort
```

У DevTools перевірте розміри 360×800, 768×1024, 1440×900 та 844×390. На кожному відкрийте форму й підтвердження. Перевірте також ширини 639 і 640 px для зміни розташування кнопок. Для довгого вмісту тимчасово додайте до основного слота кілька абзаців, які перевищують висоту панелі. Прокручується лише основний вміст; заголовок і кнопки лишаються доступними. За масштабу 200% перевірте перенесення тексту й доступність дій.

Зробіть знімки обох сценаріїв, чотирьох розмірів екрана й довгого вмісту. Один знімок може підтверджувати кілька пунктів. Додайте вже отриманий знімок DOM. Перевірте відсутність помилок консолі.

Зупиніть dev-сервер, виконайте збірку й перевірте її через preview:

```
npm run build
npm run preview -- --port 4173 --strictPort
```

Відкрийте `http://localhost:4173` і повторіть відкриття, збереження та скасування. Зупиніть preview через Ctrl+C. Заповніть README й збережіть роботу у власному репозиторії.

Що здається

Проєкт, конфігурація Tailwind/PostCSS, знімки двох сценаріїв і адаптивних розмірів, результат збірки й пояснення ref та слотів.

Обов’язкові загальні вимоги:

- файл звіту завантажується до СЕЗН Moodle у форматі PDF;
- звіт містить титульний аркуш і деталі виконання роботи;
- звіт містить посилання на репозиторій GitHub, Bitbucket або GitLab із роботою в каталозі `lab1`, `lab2`, … за номером роботи;

- звіт містить команди, які виконуються під час роботи;
- звіт містить знімки екрана, з яких зрозумілий контекст роботи та видно персональні дані автора: адресний рядок, дані студента та відомості про середовище.

Вимоги до змісту та оформлення звіту

- Файл звіту завантажується до СЕЗН Moodle у форматі PDF.
- Звіт містить титульний аркуш і деталі виконання роботи.
- Звіт містить посилання на репозиторій GitHub, Bitbucket або GitLab із роботою в каталозі `lab1`, `lab2`, … за номером роботи.
- Звіт містить команди, які виконуються під час роботи.
- Звіт містить знімки екрана, з яких зрозумілий контекст роботи та видно персональні дані автора: адресний рядок, дані студента й відомості про середовище.

Крім переліченого, за цією роботою здається:

- Вихідний Vue-проєкт у каталозі lab3 для денної або lab2 для заочної форми, конфігурація Tailwind/PostCSS та package-lock.json
- README зі структурою компонентів, версіями, командами запуску й збірки та описом контракту ModalDialog
- Таблиця перевірок з очікуваним і фактичним результатом, зокрема скасування, валідації, фокуса, видалення та повторного відкриття
- Знімки обох застосувань вікна, чотирьох розмірів екрана, довгого вмісту та DOM із Teleport; один знімок може підтверджувати кілька перевірок
- Результат npm run build і коротке пояснення різниці між реактивним ref, шаблонним ref та слотами

Контрольні питання

1. Чим пропс title відрізняється від слота за замовчуванням та іменованого слота footer?

2. Чим реактивний ref відрізняється від шаблонного ref і коли DOM-посилання може бути null?

3. Що змінює Teleport і чи змінюються після нього пропси, події та контекст слотів?

4. Чому фон відновлюють у after-leave, а не одразу після зміни isOpen на false?

5. Як зберегти можливість скасування редагування і куди повернути фокус, якщо картку видалено?

Література до роботи

1. Vue.js. Component v-model — https://vuejs.org/guide/components/v-model.html
2. Vue.js. Slots — https://vuejs.org/guide/components/slots.html
3. Vue.js. Reactivity Fundamentals — https://vuejs.org/guide/essentials/reactivity-fundamentals.html
4. Vue.js. Template Refs — https://vuejs.org/guide/essentials/template-refs.html
5. Vue.js. Teleport — https://vuejs.org/guide/built-ins/teleport.html
6. Vue.js. Transition — https://vuejs.org/guide/built-ins/transition.html
7. Tailwind CSS 3. Installation with Vite — https://v3.tailwindcss.com/docs/guides/vite#vue
8. Tailwind CSS 3. Responsive Design — https://v3.tailwindcss.com/docs/responsive-design
9. Tailwind CSS 3. Reduced motion — https://v3.tailwindcss.com/docs/hover-focus-and-other-states#prefers-reduced-motion
10. WAI-ARIA APG. Dialog Modal Pattern — https://www.w3.org/WAI/ARIA/apg/patterns/dialog-modal/
11. MDN. inert — https://developer.mozilla.org/en-US/docs/Web/HTML/Reference/Global_attributes/inert