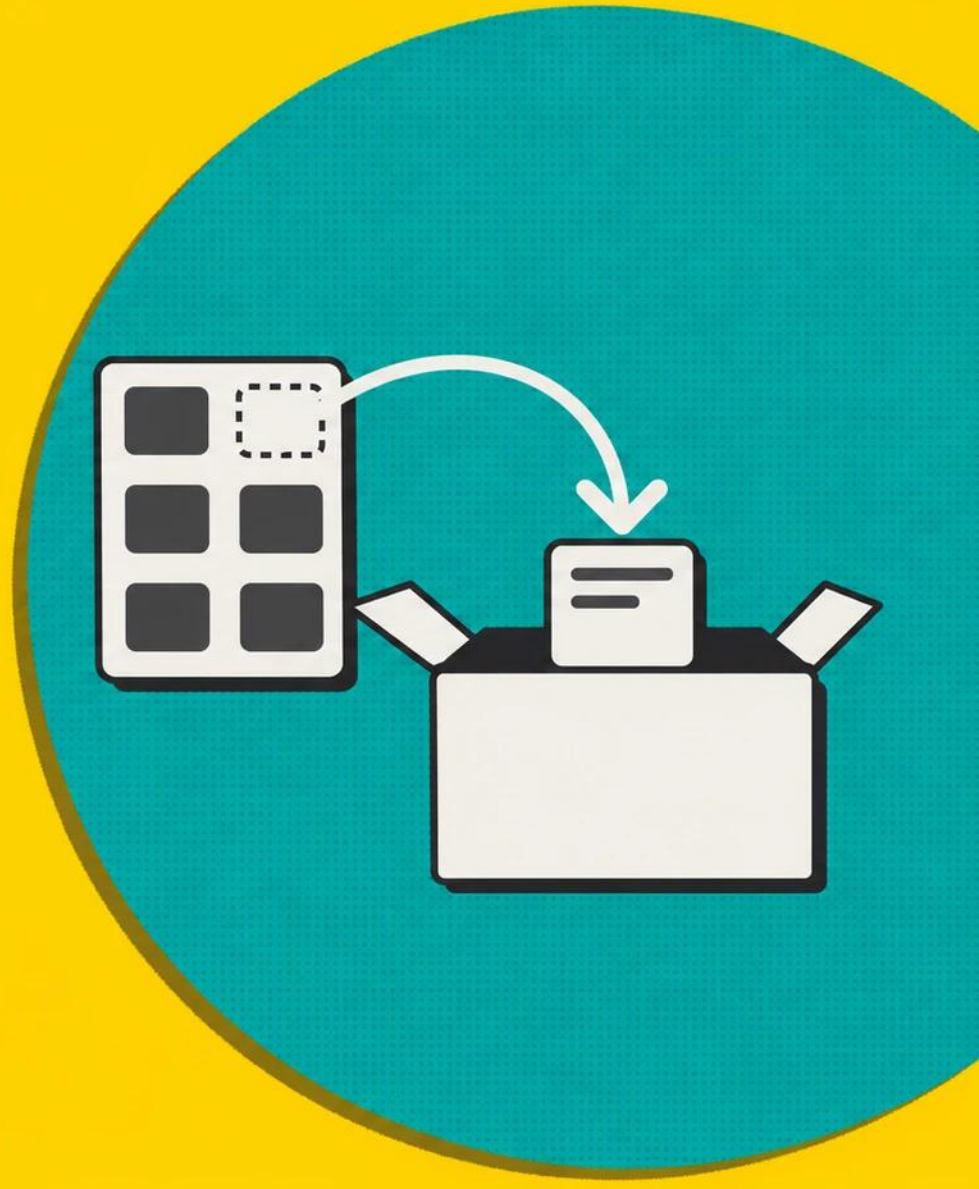
1.11

TEMPS 1 • LE PORTFOLIO • SÉANCE 3

# shadcn-vue

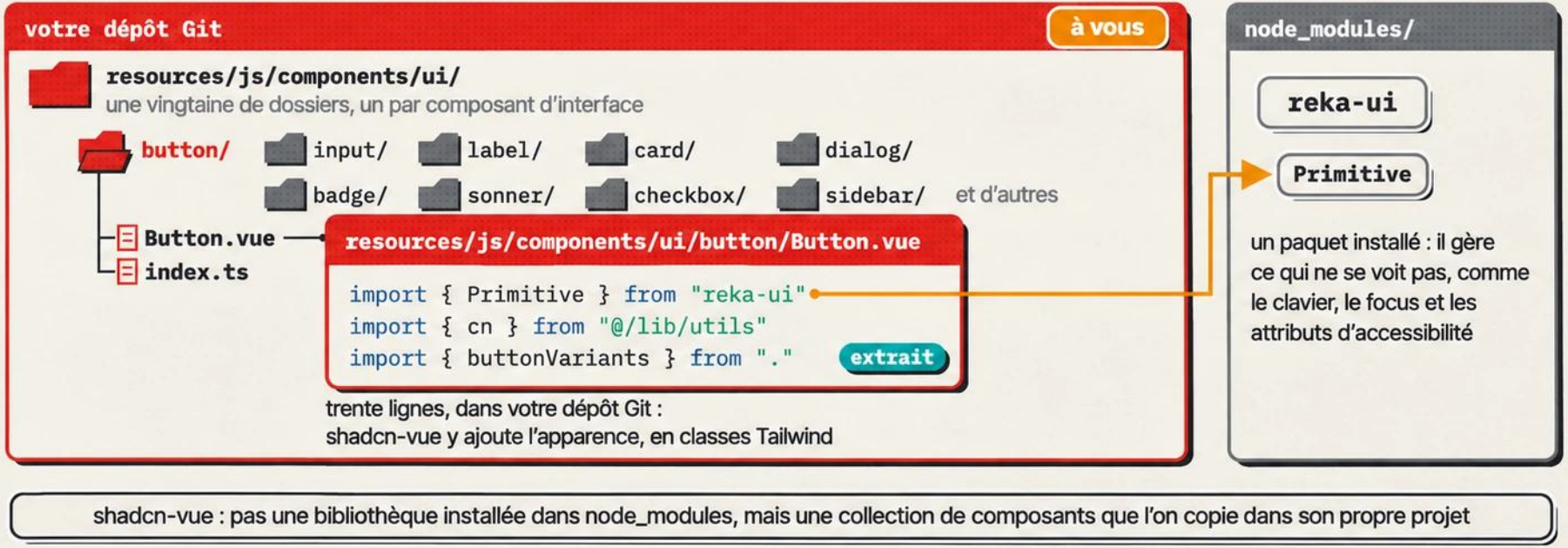
Thibault Six • IFOSUP Wavre • 2026-2027



5XDYN • Projet web dynamique • 1.11

? La page de connexion est faite de Input, de Label et de **Button**. Où se trouve le code de **Button** ?

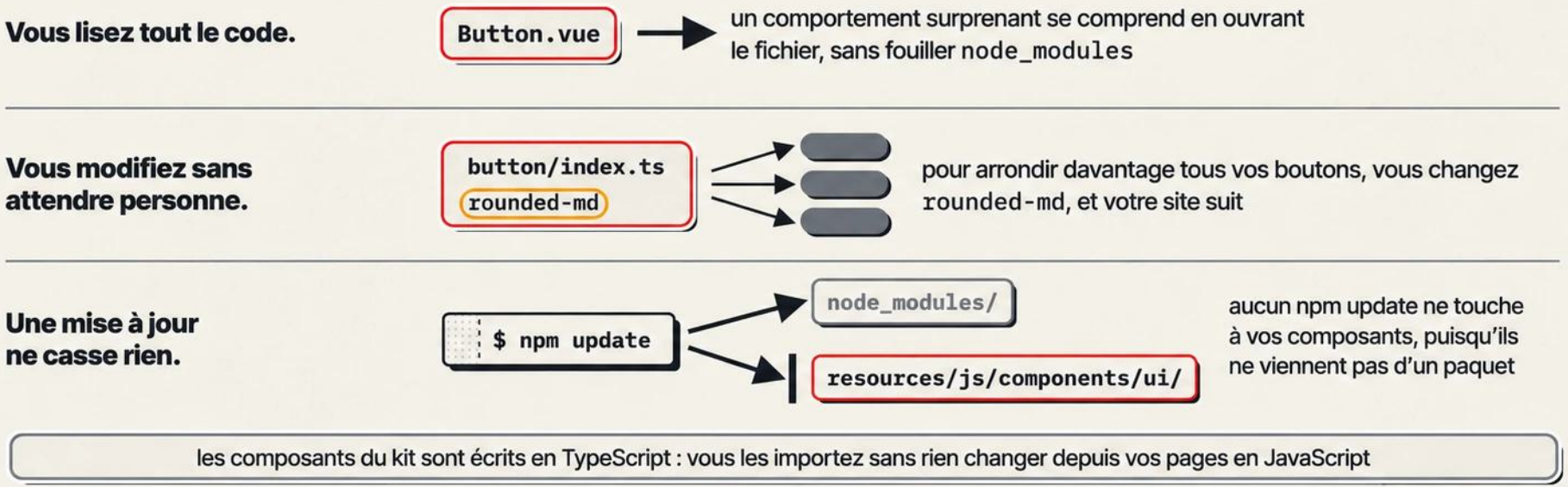
# Le code de Button est dans votre dépôt : l'apparence est à vous, le clavier et le focus viennent de reka-ui.




À gauche, le fichier que vous possédez : l'apparence du bouton. À droite, le paquet qui fait le travail difficile, que vous ne voyez pas.

? Le code du bouton est dans votre dépôt plutôt que dans un paquet. Qu'est-ce que ça change pour vous ?

# Ce code vous appartient : vous le lisez, vous le modifiez, et aucun npm update n'y touche.




Les trois rangées découlent d'un seul fait : le fichier est chez vous. Lisez chaque rangée de gauche à droite.



1.11

? En 5XCOS, alert.blade.php s'affichait par <x-alert>. Comment s'écrit le même composant en Vue ?

# Un composant Vue repose sur les trois idées de Blade : un fichier, des props, un slot ; seule la page doit l'importer.

**resources/views/components/alert.blade.php**

```
@props(['type' => 'info'])

<div class="alert alert-{{ $type }}">
    {{ $slot }}
</div>
```

les props

le slot

extrait

**resources/js/components/Alert.vue**

```
<script setup>
defineProps({
    type: { type: String, default: 'info' },
});
</script>

<template>
    <div :class="type === 'danger' ? 'text-red-700' : 'text-neutral-700'">
        <slot />
    </div>
</template>
```

extrait

DANS LA PAGE `<x-alert type="danger">` → `import Alert from '@/components/Alert.vue';` → `<Alert type="danger">`

trouvé d'après le nom du fichier

une ligne `import` dans chaque page qui l'utilise

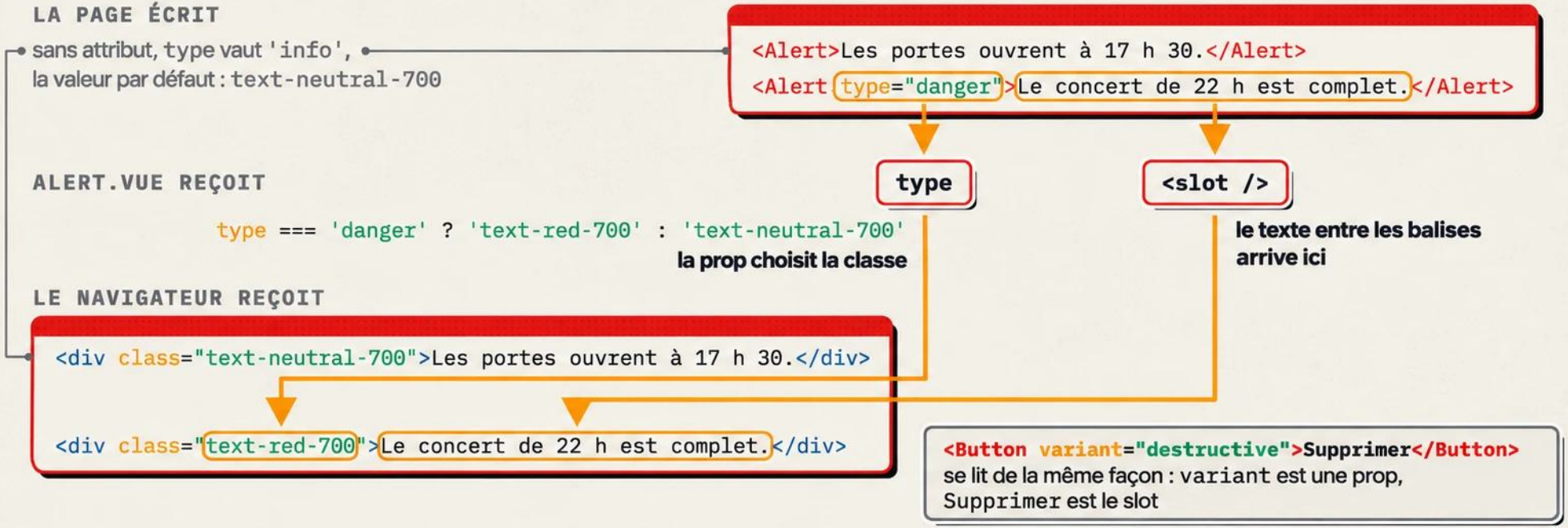
la balise porte le nom importé

le nom du fichier commence par une majuscule : `Alert.vue`, `ProjectCard.vue`

Lisez les deux fichiers idée par idée : la déclaration des props, puis le slot. La seule différence est la ligne import, que Blade n'exigeait pas.

? `<Alert type="danger">Le concert de 22 h est complet.</Alert>` : où vont l'attribut et le texte ?

# L'attribut arrive dans la prop type, le texte entre les balises arrive à la place de <slot /> : le navigateur reçoit une .



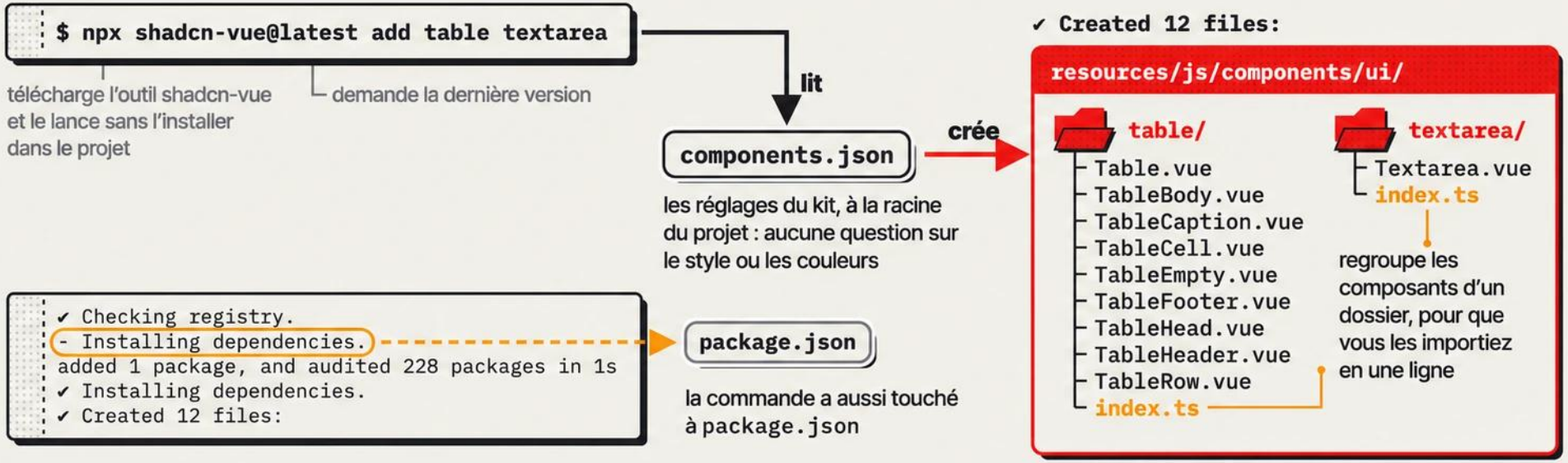

Suivez les deux flèches : l'attribut descend dans la prop, le texte descend dans le slot, et les deux ressortent dans la div. Sans attribut, la prop garde sa valeur par défaut.



1.11

? Le tableau demande Table, le formulaire du chapitre 1.12 demande Textarea, et le kit n'a ni l'un ni l'autre. Comment les ajouter ?

# La commande de shadcn-vue lit components.json, puis crée douze fichiers dans resources/js/components/ui/.




Suivez la commande : elle lit les réglages du kit, puis crée les fichiers. Le résultat se lit sur la dernière ligne : douze fichiers créés.



1.11

? « Installing dependencies » : qu'est-ce que la commande a changé d'autre dans votre projet ?

# Deux lignes de package.json ont changé : relisez ce fichier après chaque commande qui installe quelque chose.




Le diff montre ce que la commande a ajouté sans vous le demander : une version relevée, un paquet nouveau.



1.11

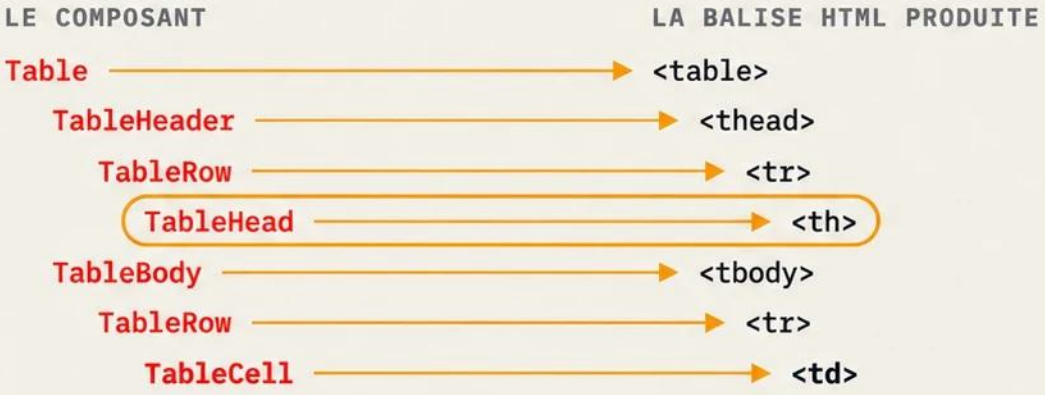
? Le dossier table/ exporte neuf composants. Pourquoi tant de morceaux pour un seul tableau ?

# Chaque morceau de Table produit sa balise HTML, avec ses classes : TableHead un <th>, TableCell un <td>.

LES ONZE COMPOSANTS DU TEMPS 1

ajoutés par la commande — neuf étaient déjà dans le kit

Button · Input · Textarea · Label · Checkbox · Spinner · Card · Table · Dialog · Badge · Toaster




et trois autres : TableCaption, TableEmpty, TableFooter

**Un composant par balise : vous écrivez la structure d'un tableau HTML, et chaque morceau y apporte les classes de shadcn-vue.**

? Au chapitre 1.12, une notification apparaîtra après l'enregistrement d'un projet. Faudra-t-il ajouter le Toaster dans chaque page ?

# Le layout du back-office place déjà un <Toaster /> : toutes ses pages peuvent afficher une notification.

**resources/js/layouts/app/AppSidebarLayout.vue**

```vue
<template>
    <AppShell variant="sidebar">
        <AppSidebar />
        <AppContent variant="sidebar" class="min-w-0 overflow-x-clip">
            <AppSidebarHeader :breadcrumbs="breadcrumbs" />
            <slot />
        </AppContent>
        <Toaster />
    </AppShell>
</template>
```

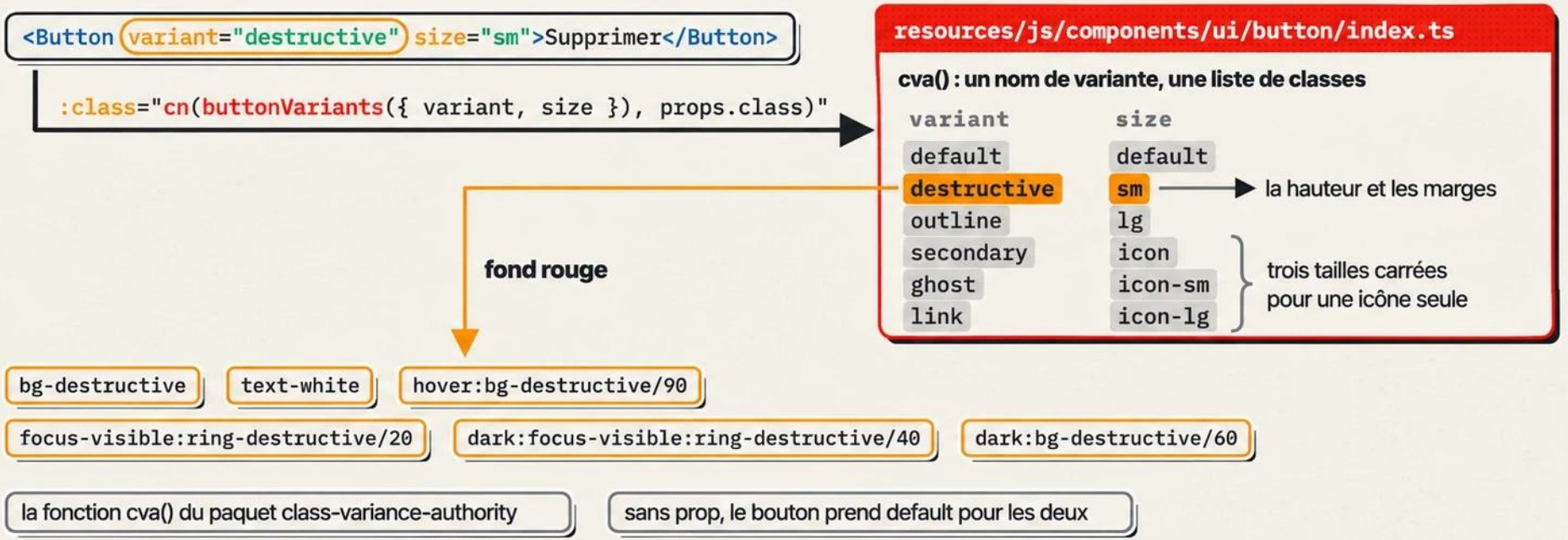



les notifications, appelées toasts, qui apparaissent dans un coin de l'écran

**Chaque ligne du template dessine une zone de l'écran. Toute page du back-office passe par ce layout : chacune peut donc afficher une notification.**

? Un bouton « Supprimer » ne ressemble pas à un bouton « Annuler ». Faut-il écrire leurs classes à chaque fois ?

# cva() associe chaque nom de variante à une liste de classes : la prop variant choisit le nom, vous n'écrivez aucune classe.




**Suivez les deux props : chaque nom mène à ses classes. Les deux se combinent sur le même bouton.**

? Nouveau projet, Supprimer, Modifier, Annuler : quelle variante choisissez-vous pour chacun ?

# Chaque variante a son usage : default pour l'action principale, destructive pour une action qui supprime.

le vrai composant Button du kit, dans le navigateur

| Nouveau projet | Supprimer | Modifier |
|---|---|---|
| `default` | `destructive` | `outline` |
| **l'action principale de la page : « Nouveau projet », « Créer le projet »** | une action qui supprime : « Supprimer » | une action secondaire dans une ligne : « Modifier » |

| Annuler | | Voir le projet |
|---|---|---|
| `secondary` | `ghost` | `link` |
| une action de retrait : « Annuler » | une icône seule | un lien au milieu d'un texte |

Badge fonctionne de la même façon : default, secondary, destructive, outline

**Choisissez la variante d'après ce que fait le bouton. Avant d'écrire des classes à la main, cherchez la variante qui existe.**

? « Nouveau projet » mène à une autre page. La balise produite doit-elle être un <button> ou un <a> ?

# Avec as-child, Button ne produit pas de <button> : il donne ses classes au <Link>, qui produit un <a>.

SANS AS-CHILD

```
<Button variant="destructive" size="sm">Supprimer</Button>
```

→ Primitive → <button>

as: "button"

AVEC AS-CHILD

resources/js/pages/admin/projects/Index.vue

```
<Button as-child>
    <Link :href="create()">Nouveau projet</Link>
</Button>
```

→ Primitive → <Link> le <Link> d'Inertia

ne produit pas de <button> : il donne ses classes à son enfant

<a href="/admin/projets/nouveau">

create() vient de Wayfinder

create() admin.projects.create /admin/projets/nouveau

un lien qui ressemble à un bouton, et qui navigue sans recharger la page

Nouveau projet

le kit utilise la même écriture dans sa barre latérale : `SidebarMenuButton as-child`

**Même composant, deux balises. Dans l'outil d'inspection du navigateur, le bouton « Nouveau projet » est une balise <a>.**

? Au chapitre 1.13, il vous faudra une boîte qui contient un formulaire. Où apprendre à vous servir de Dialog ?

# La page du composant donne la commande et des exemples : remplacez pnpm dlx par npx, et retirez lang="ts".

`https://www.shadcn-vue.com/docs/components/dialog`

- la commande d'installation
- des exemples complets
- la liste des morceaux du composant

~~`pnpm dlx`~~ → `npx`

`pnpm dlx` : un autre gestionnaire de paquets ;
npx, vous l'utilisez déjà

`<script setup lang="ts">` → `<script setup>`

les exemples sont en TypeScript : retirez `lang="ts"`
et les annotations comme : `string` si vous en trouvez

→ votre page en JavaScript

lisez un exemple avant de construire un composant à partir de rien

le kit en contient aussi un : `resources/js/components/DeleteUser.vue`

**Deux retouches suffisent pour passer de la documentation à votre projet. Gardez la structure de l'exemple, mettez-y vos données.**

1.11

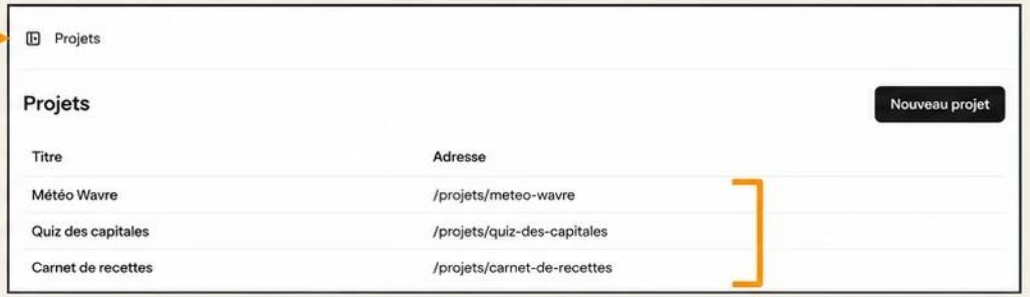
? Dans Index.vue, quel code produit le fil d'Ariane « Projets », et quel code produit une ligne par projet ?

# Index.vue assemble les morceaux : defineOptions donne le fil d'Ariane au layout, TableRow répète une ligne par projet.

**resources/js/pages/admin/projects/Index.vue** — extrait

```
defineOptions({
    layout: {
        breadcrumbs: [{ title: 'Projets', href: index() }],
    },
});
```

`breadcrumbs` : la liste des étapes du fil d'Ariane, chacune avec un titre et une adresse




`/admin/projets` à la fin du chapitre

**resources/js/pages/admin/projects/Index.vue** — extrait

```
<TableRow v-for="project in projects" :key="project.id">
    <TableCell class="font-medium">{{ project.title }}</TableCell>
    <TableCell>/projets/{{ project.slug }}</TableCell>
</TableRow>
```

une ligne par projet, dans l'ordre de position

`:key` reçoit `project.id`, qui ne change jamais

**Suivez chaque flèche du code à l'écran. Le clic sur « Nouveau projet » mène à une page d'erreur : create() arrive au chapitre 1.12.**



1.11

? Qu'est-ce qui doit être vrai à la fin ?

# À vous, quinze minutes : le tableau de vos projets, fait de composants rangés dans votre dépôt.

## À FAIRE

1. Ajoutez Table et Textarea depuis le dossier du projet : `npx shadcn-vue@latest add table textarea`.
2. Lisez `package.json` : `git diff package.json`.
3. Remplacez la liste brute du chapitre 1.10 par un tableau, dans `resources/js/pages/admin/projects/Index.vue`.

## À VÉRIFIER

15 min

- [ ] `/admin/projets` affiche un tableau de deux colonnes avec vos trois projets, dans l'ordre de position.
- [ ] Le fil d'Ariane affiche « Projets ».
- [ ] Le bouton « Nouveau projet » a l'apparence d'un bouton, et l'outil d'inspection du navigateur montre une balise `<a href="/admin/projets/nouveau">`.
- [ ] `resources/js/components/ui/` contient les dossiers table et textarea, et `package.json` contient `cn`.

Trois étapes à gauche, quatre vérifications à droite ; la troisième se fait avec l'outil d'inspection du navigateur.