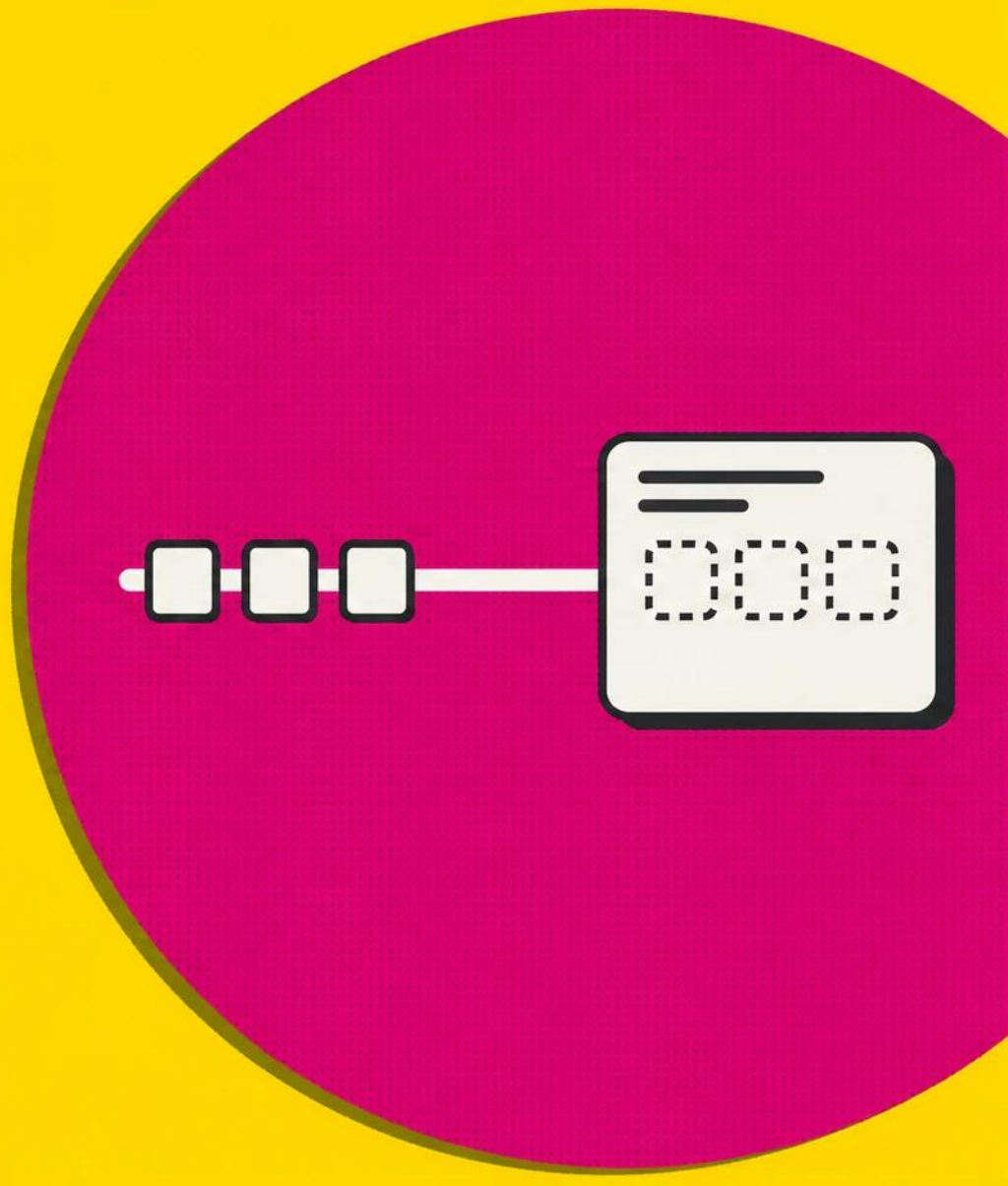
1.07

TEMPS 1 · LE PORTFOLIO · SÉANCE 2

Pages et props

Thibault Six · IFOSUP Wavre · 2026-2027



? Cette fiche affiche trois champs. Qu'est-ce que le visiteur peut lire en plus ?

La fiche n'affiche que le titre, le résumé et le contenu ; le contrôleur a pourtant passé le modèle entier.

ProjectController.php

```php
public function show(Project $project): Response
{
    return Inertia::render('projects/Show', [
        'project' => $project,
    ]);
}
```

?

qu'a reçu le navigateur en plus ?

/projets/quiz-des-capitales

Quiz des capitales

Un quiz de géographie en Vue.js avec score, minuteur et classement local.

en binôme, cinq minutes

indice : au chapitre 1.04, vous avez trouvé `author` dans le code source de la page

Lisez le contrôleur, puis l'écran : trois champs affichés, un modèle entier passé. Cherchez l'écart.

? La page montre trois champs. Qu'a reçu le navigateur ?

Tout ce que le contrôleur passe à la page arrive, lisible, dans le HTML.

ProjectController.php · serveur

```
return Inertia::render('projects/Show', [
    'project' => $project,
]);

la version à corriger
le modèle entier, 13 colonnes
```

HTML

reçu par le navigateur Ctrl+U

```
<script data-page="app" …>
"project": {
  "id"            "title"
  "slug"          "summary"
  "cover_path"    "content"
  "demo_url"      "position"
  "repo_url"      "created_at"
  "published_at"  "updated_at"
}
```

reçues, jamais affichées

Vue

l'écran du visiteur

/projets/quiz-des-capitales

Quiz des capitales

Un quiz de géographie en Vue.js avec score, minuteur et classement local.

le code source de la fiche, Ctrl+U, recadré

{"id":2,"title":"Quiz des capitales","slug":"quiz-des-capitales","summary":"Un quiz
"position":2,"created_at":"2026-10-29T16:55:01.000000Z","updated_at":"2026-10-

En orange : ce que tout visiteur peut lire sans que la page l'affiche. Une prop n'est jamais secrète.

1.07

? Comment garder six colonnes d'un projet sans les recopier une à une ?

only() ne garde que les colonnes que vous nommez : un tableau de six clés, rien du reste.

```
$project->only('title', 'slug', 'summary', 'content', 'demo_url', 'repo_url')
```

six noms demandés, six clés rendues → un tableau PHP de six clés

le modèle Project

id	title	slug	summary	content
cover_path	demo_url	repo_url	published_at	
position	created_at	updated_at	tags	

sept colonnes que la page n'affiche pas

```
'title' => 'Quiz des capitales'
'slug' => 'quiz-des-capitales'
'summary' => 'Un quiz de géographie…'
'content' => '<p>Projet du cours …</p>'
'demo_url' => 'https://example.com/quiz'
'repo_url' => 'https://github.com/exemple/quiz-capitales'
```

```
$ php artisan tinker --execute="echo json_encode(
  App\Models\Project::firstWhere('slug', 'quiz-des-capitales')
  ->only('title', 'slug', 'demo_url'), JSON_UNESCAPED_SLASHES);"
{"title":"Quiz des capitales","slug":"quiz-des-capitales","demo_url":"https://example.com/quiz"}
```

un tableau, plus un modèle : only() ne rend que des valeurs

Lisez de gauche à droite : treize colonnes, six noms entre les parenthèses, six clés dans la boîte. Le ticket Tinker le montre sur trois colonnes : only() rend un tableau PHP ordinaire, pas un modèle.

? Comment mettre ces six clés dans le tableau de la page, à côté des deux clés calculées ?

Les trois points vident la boîte de only() dans la vôtre ; sans eux, une boîte dans la boîte.

ProjectController.php · show()

```
'project' => [
    ...$project->only('title', 'slug', 'summary', 'content', 'demo_url', 'repo_url'),
    'cover_url' => $project->cover_path
        ? Storage::disk('public')->url($project->cover_path)
        : null,
    'tags' => $project->tags->pluck('name'),
],
```



SANS LES TROIS POINTS



3 clés : 0, cover_url, tags

~~project.title~~

project[0].title

AVEC LES TROIS POINTS

'project' => […]

8 clés, toutes au même niveau

project.title




```
$ php artisan tinker --execute="echo json_encode([...['a' => 1, 'b' => 2], 'c' => 3]);"   → {"a":1,"b":2,"c":3}
$ php artisan tinker --execute="echo json_encode([['a' => 1, 'b' => 2], 'c' => 3]);"      → {"0":{"a":1,"b":2},"c":3}
```

Lisez le code en entier, puis les deux volets : avec les trois points, les six clés arrivent au même niveau que les deux vôtres ; sans eux, la page recevrait une boîte dans la boîte. Le ticket le montre sur deux tableaux de rien.

? Comment construire ce tableau pour chaque projet de la liste ?

index() lit les projets triés et transforme chacun en un tableau de cinq clés.

ProjectController.php · index()

```php
1 $projects = Project::with('tags')
2     ->orderBy('position')
3     ->get()
4     ->map(fn (Project $project) => [
          'title' => $project->title,
          'slug' => $project->slug,
          'summary' => $project->summary,
          'cover_url' => $project->cover_path
              ? Storage::disk('public')->url($project->cover_path)
              : null,
          'tags' => $project->tags->pluck('name'),
      ]);
```

1. lit les projets et charge leurs étiquettes en une requête de plus
2. trie selon la colonne position, remplie par votre seeder
3. lance la requête et renvoie une collection de modèles
4. transforme chaque modèle en un tableau de cinq clés

deux clés ne recopient pas une colonne

`Storage::disk('public')->url()` : APP_URL, puis `/storage/`, puis le chemin

JSON · un projet

```
{"title":"Quiz des capitales","slug":"quiz-des-capitales",
"summary":"Un quiz de géographie…","cover_url":"http://…/quiz-capitales.jpg",
"tags":["Vue.js","JavaScript"]}
```

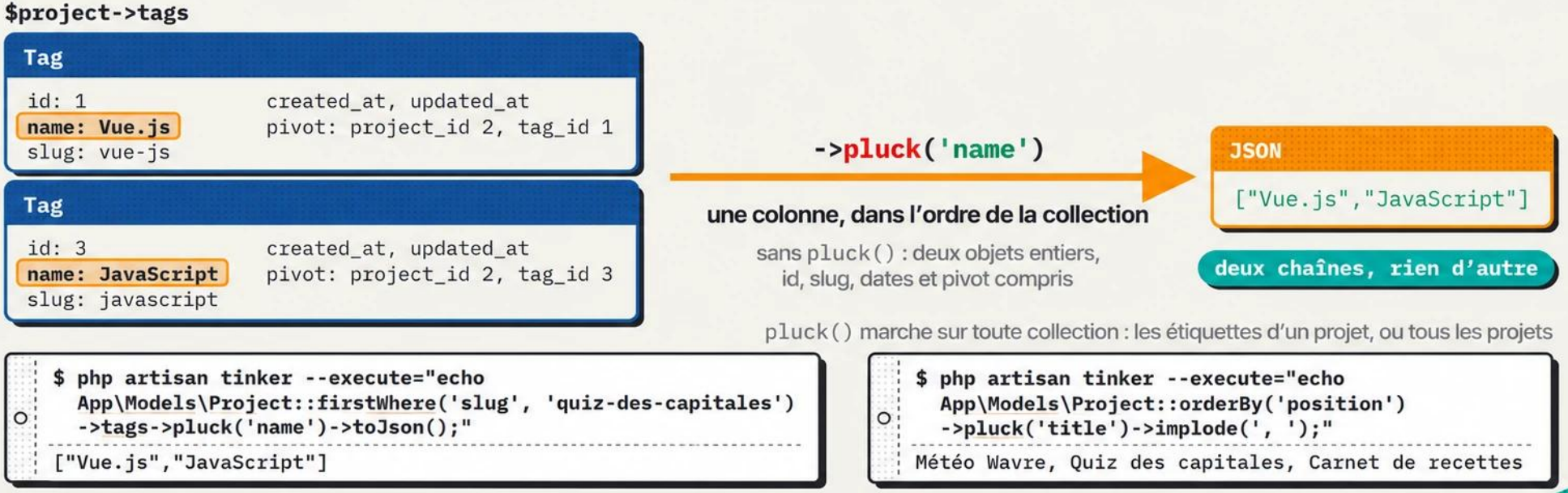
```php
Inertia::render('projects/Index',
    ['projects' => $projects])
```

`projects/Index` désigne `resources/js/pages/projects/Index.vue`

Suivez les quatre numéros de haut en bas, puis lisez le JSON d'un projet : cinq clés, dont deux calculées.

? La page affiche « Vue.js, JavaScript ». Que lui envoyer : les étiquettes entières, ou leurs noms ?

pluck('name') tire une colonne de la collection : des noms, pas des objets Tag entiers.



pluck() marche sur toute collection : les étiquettes d'un projet, ou tous les projets

```
$ php artisan tinker --execute="echo
  App\Models\Project::firstWhere('slug', 'quiz-des-capitales')
  ->tags->pluck('name')->toJson();"
["Vue.js","JavaScript"]
```

```
$ php artisan tinker --execute="echo
  App\Models\Project::orderBy('position')
  ->pluck('title')->implode(', ');"
Météo Wavre, Quiz des capitales, Carnet de recettes
```

Lisez les deux cartes, puis la flèche : pluck() prend la colonne name de chaque étiquette et rend une liste.
La page reçoit deux mots, pas deux objets avec leurs dates.

? Comment Laravel trouve-t-il le projet de l'adresse /projets/quiz-des-capitales ?

{project:slug} fait chercher le projet par son slug ; un slug inconnu répond par une 404.

`/projets/quiz-des-capitales` → **routes/web.php**

```
Route::get('projets/{project:slug}',
    [ProjectController::class, 'show'])
    ->name('projects.show');
```

la colonne de recherche n'est plus id mais slug

projects

id	slug
1	meteo-wavre
2	**quiz-des-capitales ✓ trouvé**
3	carnet-de-recettes

`/projets/nexiste-pas` → **aucune ligne** → **404** sans une ligne de code de votre part

show(Project $project)

reçoit le projet déjà trouvé

```
$ php artisan route:list --path=projets
GET|HEAD projets .. projects.index › ProjectController@index
GET|HEAD projets/{project:slug} .. projects.show › ProjectController@show
```

Suivez le slug de l'adresse à la table : Laravel cherche la ligne, la trouve, la passe à show(). Pas de ligne, 404.

1.07

? Qu'est-ce que le visiteur lit maintenant dans le code source de la fiche ?

Huit clés au lieu de treize dans le code source : rien ne fuit, et la page reçoit des données prêtes.

ProjectController.php · show()

```php
public function show(Project $project): Response
{
    $project->load('tags');
    return Inertia::render('projects/Show', [
        'project' => [
            ...$project->only('title', 'slug', 'summary', 'content', 'demo_url', 'repo_url'),
            'cover_url' => $project->cover_path
                ? Storage::disk('public')->url($project->cover_path)
                : null,
            'tags' => $project->tags->pluck('name'),
        ],
    ]);
}
```

le projet est déjà lu : `load()` charge ses étiquettes après coup

avec 'project' => $project


```
"project":{"id":2,"title":…,"slug":…,"summary":…,"content":…,
"cover_path":…,"demo_url":…,"repo_url":…,"published_at":…,
"position":…,"created_at":…,"updated_at":…}
```


avec votre tableau


```
"project":{"title":"Quiz des capitales","slug":"quiz-des-capitales","summary":"…","content":"<p>Projet du
cours …<\/p>","demo_url":"https:\/\/example.com\/quiz","repo_url":"https:\/\/github.com\/exemple\/
quiz-capitales","cover_url":"http:\/\/…\/quiz-capitales.jpg","tags":["Vue.js","JavaScript"]}
```


huit clés

Rien ne fuit par accident
une colonne ajoutée plus tard n'arrive pas tant que vous ne l'ajoutez pas au tableau

La page reçoit des données prêtes
l'adresse de l'image est calculée côté serveur

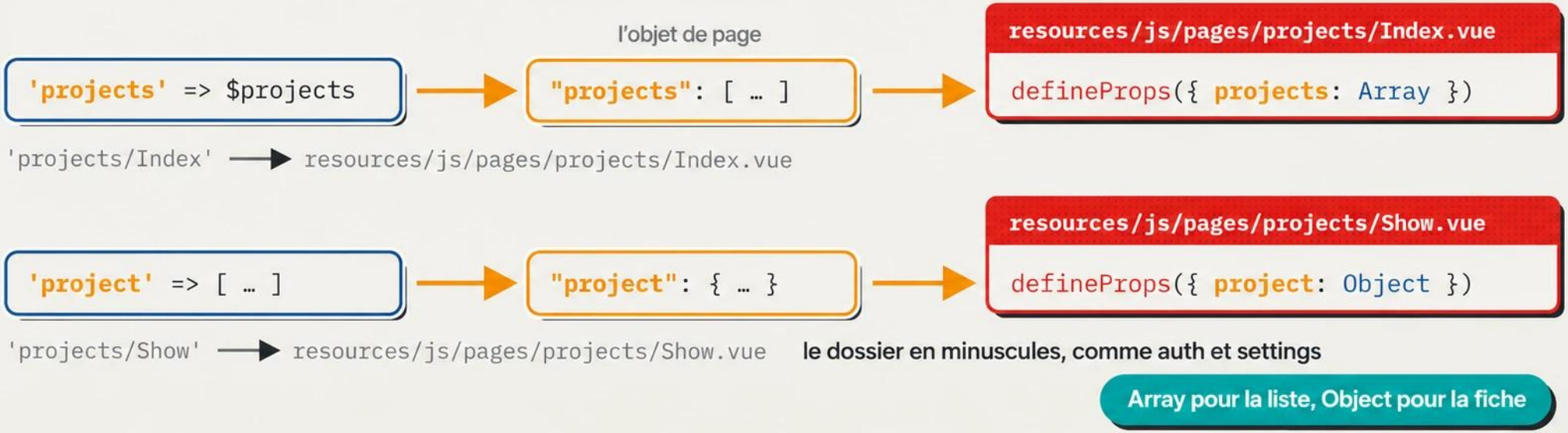
Le contrat se lit en un endroit
les clés du tableau sont exactement les props que la page déclare

Comparez les deux JSON : treize clés avant, huit après ; cherchez id, cover_path et created_at dans le second, vous ne les trouverez pas. Les trois autocollants disent pourquoi ça compte.

1.07

? Comment la page sait-elle ce qu'elle reçoit ?

La clé du tableau PHP et la prop déclarée par la page portent le même nom.



Suivez chaque nom de gauche à droite : il ne change jamais. C'est le contrat entre le contrôleur et la page.

1.07

? Comment afficher trois projets avec un seul élément li ?

v-for répète l'élément pour chaque projet, et :key lui donne un identifiant unique et stable : le slug.

projects/Index.vue · template

```
<li v-for="project in projects" :key="project.slug">
    <img v-if="project.cover_url" :src="project.cover_url" alt="" width="300" />
    <h2>{{ project.title }}</h2>
    <p>{{ project.summary }}</p>
    <a :href="`/projets/${project.slug}`">Voir le projet</a>
</li>
```

un tour par projet, dans l'ordre de position

```
:key = "meteo-wavre"
:key = "quiz-des-capitales"
:key = "carnet-de-recettes"
```

les deux points : Vue calcule la valeur au lieu de l'écrire telle quelle

v-if : pas d'image cassée pour un projet sans image

alt="" : le titre est à côté, le lecteur d'écran passe l'image

l'adresse en dur sera remplacée au chapitre 1.08

la page /projets sans style

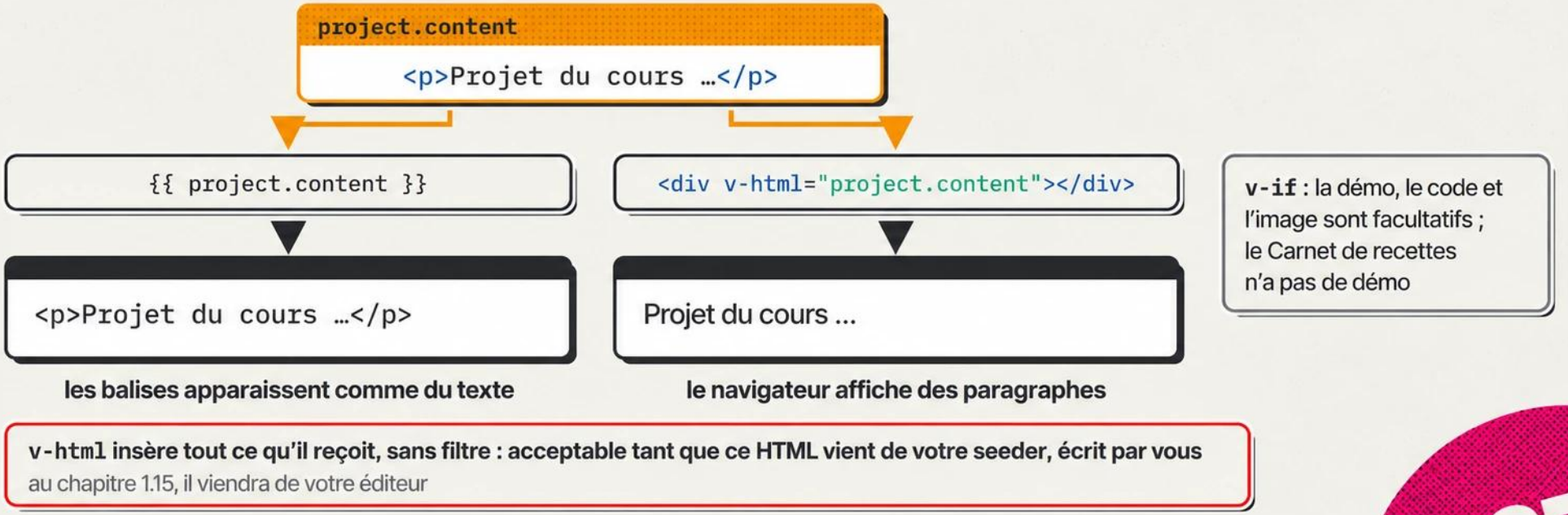
Météo Wavre Une page qui affiche la météo de Wavre heure par heure à partir d'une API publique. Voir le projet	Quiz des capitales Un quiz de géographie en Vue.js avec score, minuteur et classement local. Voir le projet

Lisez le li une fois, puis les trois cartes : même élément, trois projets, trois clés différentes et stables.

1.07

? Pourquoi {{ project.content }} n'affiche-t-il pas des paragraphes ?

Les doubles accolades affichent les balises comme du texte ; v-html insère le contenu comme du HTML, sans filtre.



v-if : la démo, le code et l'image sont facultatifs ; le Carnet de recettes n'a pas de démo

`v-html` insère tout ce qu'il reçoit, sans filtre : acceptable tant que ce HTML vient de votre seeder, écrit par vous
au chapitre 1.15, il viendra de votre éditeur

Même chaîne, deux rendus : à gauche le texte des balises, à droite les paragraphes. Le filtre, c'est vous.

? Pourquoi /projets est-elle blanche, avec l'erreur du chapitre 1.04 ?

Le résolveur inversé ne donne le layout d'administration qu'à Dashboard et admin/, et rien par défaut.

app.ts, avant

```
case name === 'Welcome' || name === 'Home':
    return null;
default:
    return AppLayout;
```

toute page inconnue reçoit le back-office :
la liste et la fiche plantent,

```
TypeError: Cannot read properties of null
(reading 'name')
```

app.ts, après

```
layout: (name) => {
    switch (true) {
        case name.startsWith('auth/'):
            return AuthLayout;
        case name.startsWith('settings/'):
            return [AppLayout, SettingsLayout];
        case name === 'Dashboard' || name.startsWith('admin/'):
            return AppLayout;
        default:
            return null;
    }
},
```

pas de layout : Home, projects/Index, projects/Show tombent ici

`switch (true)` teste chaque case dans l'ordre et s'arrête au premier qui vaut `true`

au chapitre 1.08, ce `default` recevra votre propre layout

Avant, la règle disait « back-office sauf exceptions » ; après, « rien sauf le back-office ». Lisez les case dans l'ordre.

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

À vous, trente minutes : la liste et la fiche, brutes, avec des props que vous avez choisies.

30 min

À FAIRE

1. `ProjectController` avec `index()` et `show()`, chacun avec un tableau de props construit.
2. Les deux routes, `/projets` et `/projets/{project:slug}`, nommées `projects.index` et `projects.show`.
3. Les pages `projects/Index.vue` et `projects/Show.vue`, brutes.
4. Le résolveur de `app.ts` inversé.

À VÉRIFIER

- ☐ `portfolio.test/projets` montre vos trois projets dans l'ordre de position, chacun avec son image, son titre, son résumé et un lien.
- ☐ Le lien ouvre `/projets/` suivi du slug, avec les étiquettes, le contenu en paragraphes et les liens. Un projet sans démo n'affiche pas le lien « Démo ».
- ☐ `portfolio.test/projets/nexiste-pas` répond par une erreur 404.
- ☐ Dans le code source de la fiche, cherchez `"project"` : vous ne trouvez ni `"id"`, ni `cover_path`, ni `created_at`.

Quatre étapes à gauche, quatre vérifications à droite ; la dernière se fait avec Ctrl+U, comme à la slide 2.