1.18

TEMPS 1 · LE PORTFOLIO · SÉANCE 5

La page publique

Thibault Six · IFOSUP Wavre · 2026-2027



? Votre planche fixe une couleur et une police. Où les écrire pour les changer en une ligne ?

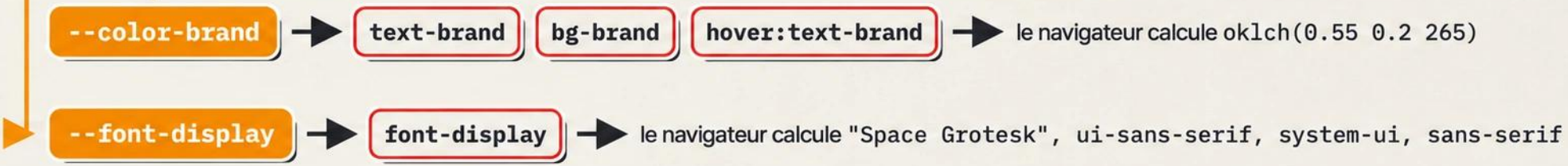
Chaque variable de @theme est un jeton : --color-brand donne bg-brand, --font-display donne font-display.

juste après la ligne `@plugin` du chapitre 1.15

resources/css/app.css

```
@theme {
    --color-brand: oklch(0.55 0.2 265);
    --font-display: 'Space Grotesk', ui-sans-serif, system-ui, sans-serif;
}
```

le préfixe choisit la famille de classes



les valeurs de l'exemple : un indigo, et Space Grotesk pour les titres. Remplacez-les par celles de votre planche.

Lisez de gauche à droite : le préfixe de la variable choisit la famille, la fin de son nom devient la fin de la classe.

? Votre planche parle d'une couleur d'accent. Pourquoi ne pas appeler le jeton accent ?

Le kit se sert déjà d'accent dans tout le back-office : nommez votre jeton brand, qui garde la même valeur en mode sombre.

UN NOM QUE LE KIT N'UTILISE PAS

plus bas, le bloc `@theme inline` du kit

resources/css/app.css

```
--color-accent: var(--accent);
--color-accent-foreground: var(--accent-foreground);
```

un gris clair : le survol des boutons ghost et outline, la barre latérale

redéfinir `accent` repeindrait tout le back-office

déjà pris, de la même façon :

`primary` `secondary` `muted` `destructive` `border` `ring`

brand ✓ **un nom à vous**

UNE COULEUR FIXE, MÊME EN MODE SOMBRE

votre jeton



le kit



déclarée deux fois

relie `--color-accent` à cette variable

votre planche prévoit une autre couleur sur fond sombre ? Recopiez cette forme pour `brand`. Sinon, `@theme` suffit.

À gauche, le nom à éviter et pourquoi. À droite, brand garde une seule valeur, alors que le kit déclare deux fois la sienne.

? Le kit charge déjà une police. Où ajouter la vôtre, et d'où vient la classe font-display ?

Ajoutez la police à fonts, dans vite.config.ts, avec bunny(), et --font-display en fait la classe font-display.

vite.config.ts

```
fonts: [
    bunny('Instrument Sans', {
        weights: [400, 500, 600],
    }),
    bunny('Space Grotesk', {
        weights: [500, 700],
    }),
],
```

la police du kit

la vôtre, ajoutée dans la liste

les graisses notées sur votre planche

LE TERMINAL DE COMPOSER RUN DEV

```
vite.config.ts changed, restarting server..
server restarted.
```

vous enregistrez : Vite redémarre tout seul

déclarée dans @theme, slide 1

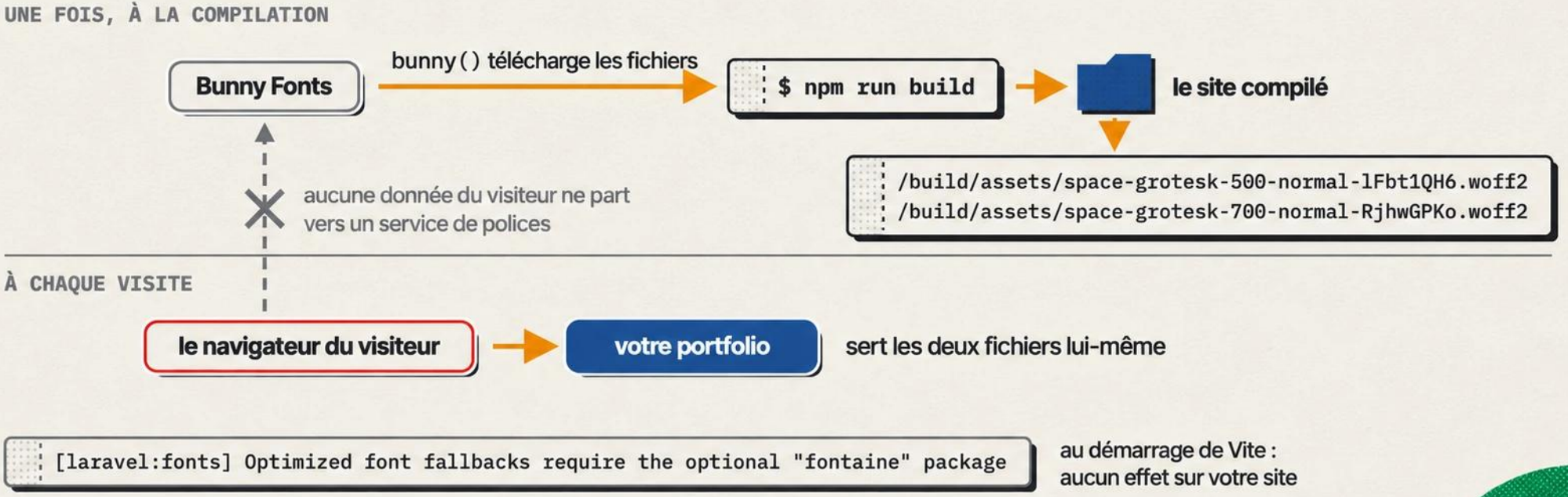
`--font-display` → `font-display`

la police est chargée, la classe l'applique

Suivez la ligne : vous ajoutez la police, Vite redémarre, et la classe font-display l'applique.

? Avec bunny(), le navigateur du visiteur contacte-t-il Bunny Fonts à chaque visite ?

Non : bunny() télécharge les fichiers, npm run build les copie dans le site compilé, et votre site les sert lui-même.



En haut, ce qui se passe une fois, à la compilation. En bas, chaque visite : les fichiers viennent de votre site.

1.18

? La route / n'envoie que votre nom. Que lui manque-t-il pour répondre au lecteur pressé ?

La route / appelle désormais HomeController, un contrôleur invocable : Laravel en exécute la méthode __invoke().

LES TROIS QUESTIONS DU LECTEUR PRESSÉ, CHAPITRE 1.01

qui vous êtes | ce que vous avez construit | comment vous écrire

il faut vos projets publiés, plus seulement votre nom

DEPUIS LE CHAPITRE 1.04

la route / appelle une fonction anonyme

MAINTENANT

```
use App\Http\Controllers\HomeController;
```

en tête du fichier

routes/web.php

```
Route::get('/', HomeController::class)->name('home');
```

le nom home ne change pas : le lien de votre nom mène toujours à l'accueil

app/Http/Controllers/HomeController.php

```
class HomeController extends Controller
{
    public function __invoke(): Response
```

extrait

invocable : la route nomme la classe sans méthode, et Laravel appelle `__invoke()`

En haut, ce que l'accueil doit dire. En bas, la route passe d'une fonction anonyme à un contrôleur d'une seule méthode.

1.18

? Parmi vos projets en base, lesquels arrivent sur l'accueil, et avec quelles clés ?

HomeController garde les trois premiers projets publiés, par position, et n'envoie à Home que quatre clés par projet.



Suivez la requête : filtrer les publiés, trier, en garder trois, réduire chacun à quatre clés, puis envoyer à la page.

? L'accueil et la liste des projets montrent les mêmes cartes. Faut-il les écrire deux fois ?

Non : ProjectCard.vue reçoit un projet en prop, rien d'autre ; l'accueil et la liste l'appellent dans la même boucle.



Deux pages à gauche, un composant à droite : chaque page lui passe un projet, la carte fait le reste.

? Sans attributs, une image fait sauter le texte et se charge hors de vue. Que lui ajouter ?

width et height réservent la place de l'image, loading="lazy" retarde celles du bas de page, et alt="" la dit décorative.

ProjectCard.vue · <img>

```
alt=""
width="1200"
height="675"
loading="lazy"
class="aspect-video w-full object-cover"
```

extrait

width height

le navigateur connaît le rapport de l'image avant qu'elle arrive

il réserve la place : le texte ne saute pas vers le bas

la taille affichée vient de w-full aspect-video : 308 × 173 pixels sur un écran de 800 px, dans le labo

loading="lazy"

l'image se charge quand elle approche de la partie visible

une liste de dix projets ne charge plus dix images à l'ouverture

pas sur la couverture de la fiche, en haut de page : lazy retarderait l'image la plus importante

alt=""

l'image est décorative

le titre du projet la suit et dit déjà ce qu'elle montre

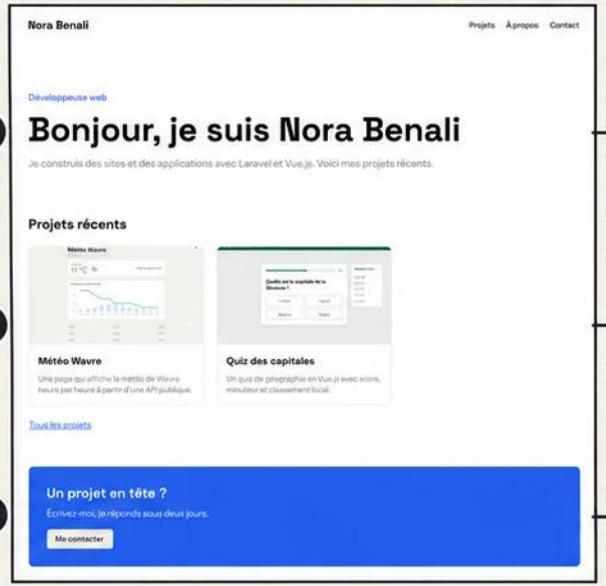
Un attribut par colonne : ce qu'il dit au navigateur, puis ce que le visiteur y gagne.

1.18

? Le lecteur pressé se pose trois questions. Comment la page d'accueil y répond-elle ?

Home.vue empile trois sections, une par question du lecteur, et la dernière est un appel à l'action qui mène au contact.

l'accueil du portfolio de l'exemple



1 qui vous êtes

```
<h1 class="font-display text-4xl font-bold md:text-6xl">
```

2 ce que vous avez construit

```
<div class="grid gap-6 sm:grid-cols-2 lg:grid-cols-3">
```

une `ProjectCard` par projet

3 comment vous écrire

```
<section class="my-16 rounded-lg bg-brand p-8 text-white">
```

un appel à l'action : un bloc qui propose au lecteur l'étape suivante

les textes sont ceux de l'exemple : écrivez ceux de votre cahier des charges

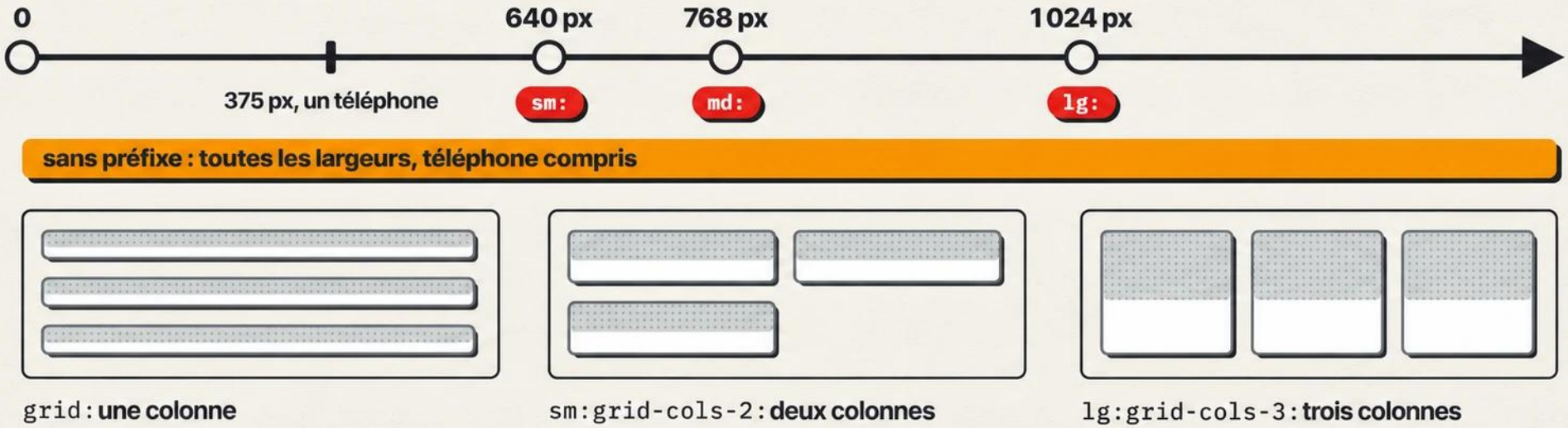
Suivez les numéros sur la capture : chaque section répond à une question, et la dernière propose de vous écrire.

? En 375 pixels, le menu déborde à côté du nom. Pour quelle largeur écrire les classes d'abord ?

Une classe sans préfixe vaut pour toutes les largeurs ; sm:, md: et lg: ajoutent ce qui change sur un écran plus large.

```
class="grid gap-6 sm:grid-cols-2 lg:grid-cols-3"
```

la grille des projets



Lisez la frise de gauche à droite : on part du téléphone, et chaque préfixe ajoute un changement à partir de sa largeur.

? Comment l'en-tête empile-t-il votre nom et le menu sur un téléphone, puis les aligne-t-il ?

Sur un téléphone, flex-col empile votre nom et le menu ; à partir de sm:, sm:flex-row les remet sur une ligne.

resources/js/layouts/PublicLayout.vue — extrait

```
<header
    class="flex flex-col gap-3 py-6 sm:flex-row sm:items-center sm:justify-between"
>
```

sans préfixe : le téléphone

à partir de 640 px : une ligne, les deux écartés

EN 375 PIXELS

Nora Benali
Projets À propos Contact

le nom au-dessus du menu

SUR UN ÉCRAN PLUS LARGE

Nora Benali Projets À propos Contact

le nom et le menu sur une ligne

un nombre plus grand signale un élément plus large que l'écran

LA CONSOLE, EN MODE APPAREIL À 375 PIXELS

```
> document.documentElement.scrollWidth
375
```

Le même en-tête deux fois : empilé sur le téléphone, sur une ligne à partir de 640 pixels. La console confirme qu'il ne déborde plus.

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

À vous, quarante-cinq minutes : votre planche appliquée aux pages publiques, lisible sur un téléphone et en mode sombre.

À FAIRE

1. Les couleurs de votre planche dans `@theme`, sous des noms que le kit n'utilise pas.
2. Vos polices, deux au plus, chargées par `bunny()` dans `vite.config.ts` et déclarées dans `@theme`.
3. `HomeController` et un accueil qui dit qui vous êtes, montre trois projets publiés et mène au contact.
4. `ProjectCard.vue` dans l'accueil et la liste ; prose sur la fiche et l'à-propos.
5. `PublicLayout.vue` écrit pour le téléphone d'abord.

À VÉRIFIER

45 min

- [ ] Chaque image a `width` et `height`, et chaque image hors du haut de page a `loading="lazy"`.
- [ ] Les tailles de vos titres suivent l'échelle de votre planche.
- [ ] À 375 pixels de large, `document.documentElement.scrollWidth` vaut 375 sur chaque page publique.
- [ ] Vos pages restent lisibles avec `prefers-color-scheme: dark`.
- [ ] `php artisan test` reste vert.

Cinq étapes à gauche, cinq vérifications à droite ; la quatrième se fait dans le panneau Rendering de Chrome.