The Nice WordCamp

Les compositions WordPress, au-delà du copier-coller

WC NICE - 2026

Et si on livrait des sites qu'on aimerait utiliser ?

Les compositions sont un des leviers pour y arriver

Architecte WP

Direction technique

SEO

Thierry **Pigot**

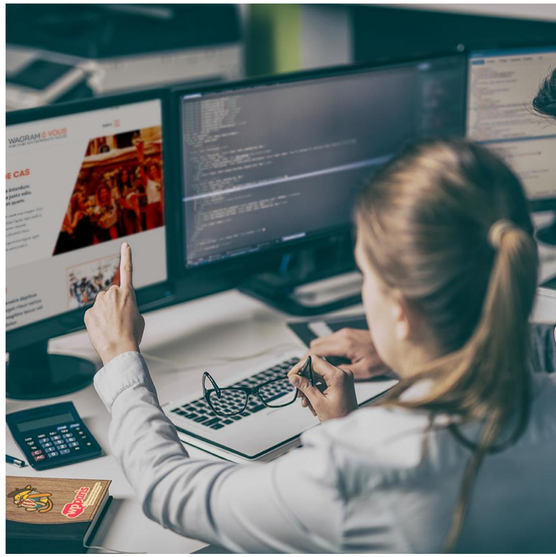
weare[wp]

WP.ASSISTANCE
Support & Maintenance WordPress

www.wearewp.pro

Sommaire

- ✓ Introduction
- ✓ PHP comme super-pouvoir
- ✓ Industrialiser sa pratique
- ✓ Contrôler l’expérience d’édition
- ✓ Q&A



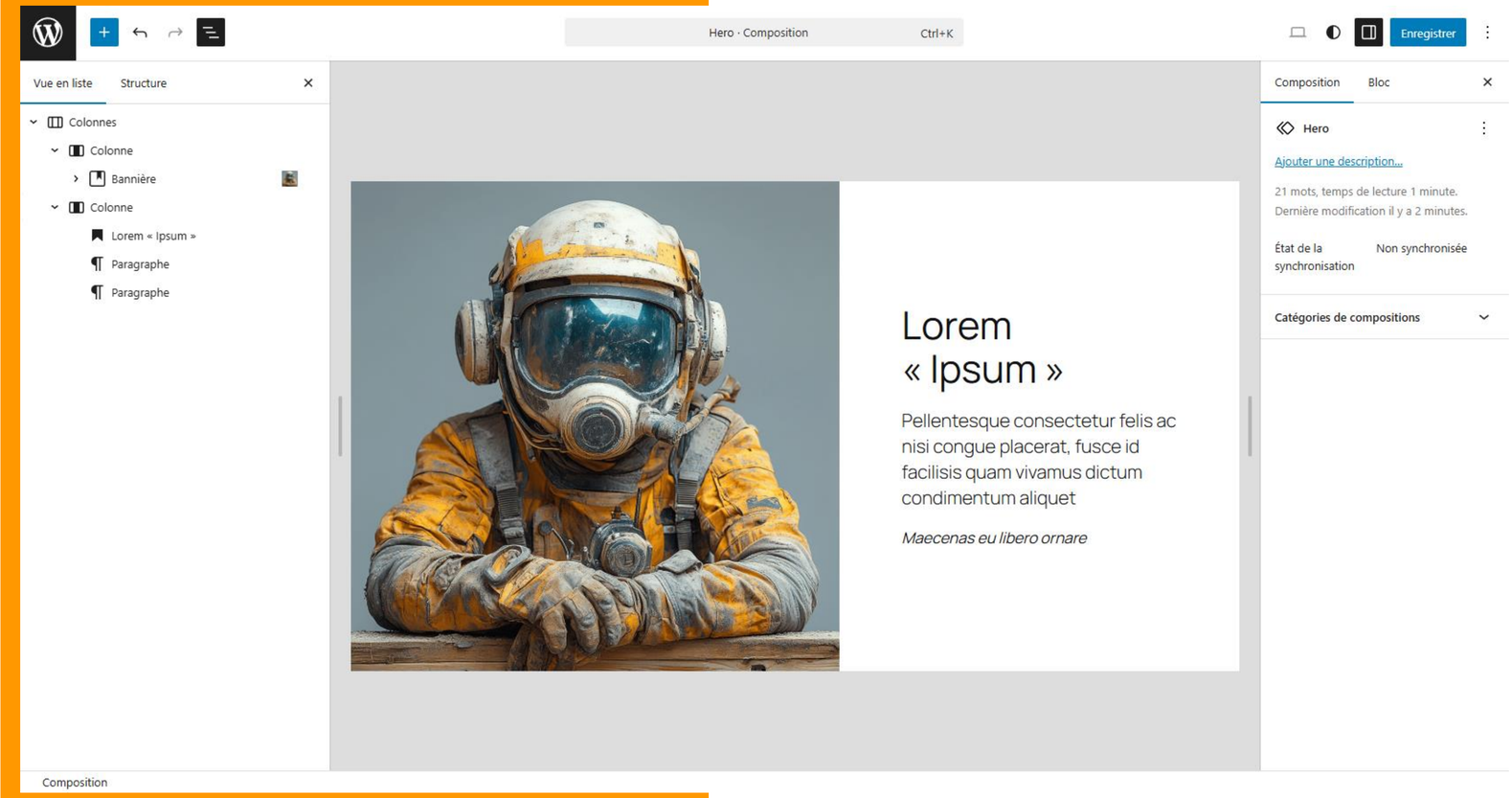
Introduction

Le workflow classique

CE QUE TOUT LE MONDE CONNAÎT

1. Aller dans **Apparence > Editeur > compositions**
2. Cliquer sur le bouton **+** (Ajouter une composition)
3. Assembler des blocs dans l'éditeur
4. Copier via **Options (⋮) > Copier tous les blocs**
5. Coller dans **/patterns/ma-composition.php**
6. **Ajouter le header** de fichier

Le workflow classique




```
<?php
/**
 * Title: Hero
 * Slug: themeslug/hero
 * Categories: featured
 */
?>
<!-- wp:columns
{"align":"full","style":{"spacing":{"blockGap":{"left":
"0"}}}} -->
<div class="wp-block-columns alignfull"><!--
wp:column {"width":"55%"} -->
  <!-- … blocs copiés … -->
<!-- /wp:column --></div>
<!-- /wp:columns -->
```

Introduction

Le workflow classique

CE QUE TOUT LE MONDE NE CONNAÎT PAS 😁

⚠️ WordPress met en cache la liste des patterns du thème dans un transient : **wp_theme_files_patterns-{hash}**

Comment corriger cela ?

- Activer / désactiver le theme
- Changer le n° de version du theme
- **define('WP_DEVELOPMENT_MODE', 'theme');**

La réalité

UN OUTIL DE DÉVELOPPEMENT COMPLET

"Les compositions ne sont PAS juste des blocs réutilisables. C'est un système de développement à part entière.«

	Ce qu'on croit	La réalité
Contenu	HTML statique	PHP exécutable
Workflow	Copier-coller artisanal	Système industrialisable
Utilisateur	Libre de tout casser	Expérience contrôlée

Exploiter des patterns c'est intéressant :

- pour vous : moins de code répété, maintenance simplifiée, réutilisation entre projets
- pour vos clients : Onboarding facilité, moins d'erreurs, design protégé
- pour le projet : internationalisation native, assets cohérents, évolutivité

Introduction

Ce qu'on va explorer

Partie 1 : l’utilisation du PHP

- internationalisation
- assets dynamiques
- logique PHP (boucles, conditions)

Partie 2 : industrialiser sa pratique

- organisation et conventions
- headers de fichiers avancés
- réutilisation entre projets
- contrôle programmatique

Partie 3 : l'expérience d'édition

- block Locking API
- les 3 niveaux de verrouillage
- starter patterns
- Onboarding client

L’utilisation du PHP

L’utilisation du PHP

Pourquoi c'est un game-changer ?

TEMPLATE PARTS VS PATTERNS : LA DIFFÉRENCE CRUCIALE

```
wc-nice/
├── style.css
├── theme.json
├── templates/          ← templates (index, single, page, etc.)
├── parts/              ← template parts (header, footer, etc.)
│   ├── header.html
│   └── footer.html
├── patterns/
└── functions.php
```

```
<!-- wp:template-part {"slug":"header","tagName":"header"} /-->

<!-- wp:group {"tagName":"main","layout":{"type":"constrained"}} -
<main class="wp-block-group">
    <!-- wp:pattern {"slug":"wc-nice/hero"} /-->
    <!-- wp:post-content /-->
</main>
<!-- /wp:group -->

<!-- wp:template-part {"slug":"footer","tagName":"footer"} /-->
```

L’utilisation du PHP

Pourquoi c'est un game-changer ?

TEMPLATE PARTS VS PATTERNS : LA DIFFÉRENCE CRUCIALE

Template Parts (fichiers .html dans /parts) :

```
<!-- wp:heading -->
<h2>Welcome to My Site</h2>
<!-- /wp:heading -->
```

- ❌ HTML statique uniquement
- ❌ Pas d'accès au PHP
- ❌ Texte non traduisible
- ❌ URLs en dur

Patterns (fichiers .php dans /patterns) :

```
<!-- wp:heading -->
<h2><?php esc_html_e( 'Welcome to My Site', 'wc-nice' ); ?></h2>
<!-- /wp:heading -->
```

- ✅ PHP exécutable
- ✅ Texte traduisible
- ✅ URLs dynamiques
- ✅ Logique conditionnelle

L’utilisation du PHP

⚠ ATTENTION : le contexte d'exécution

POINT CRITIQUE

Les compositions sont **compilés sur le hook init**, pas au moment du rendu.

Ce que ça signifie :

- le PHP est exécuté UNE SEULE FOIS lors de l'enregistrement du pattern
- le résultat est stocké comme HTML
- pas d'accès aux données globales de la requête

⚠ ATTENTION : le contexte d'exécution

POINT CRITIQUE

❌ CE QUI NE FONCTIONNE PAS :

```php
<?php
// ERREUR : ces fonctions n'ont pas accès aux données nécessaires sur init
if ( is_page() ) : ?>
    <!-- wp:paragraph -->
    <p><?php the_title(); ?></p>
    <!-- /wp:paragraph -->
<?php endif; ?>
```

Fonctions inutilisables dans les compositions :

- `is_home()`, `is_single()`, `is_page()` - Conditions de template
- `` `the_title()` ``, `the_content()`, `the_excerpt()` - Données du post
- `get_post()`, `get_queried_object()` - Requête globale
- `wp_get_current_user()` - Données utilisateur (pas encore défini sur init)

L’utilisation du PHP

⚠ ATTENTION : le contexte d'exécution

POINT CRITIQUE

✅ **CE QUI FONCTIONNE :**

- Fonctions d'internationalisation : `__()`, `_e()`, `esc_html_e()`, etc.
- Fonctions de thème : `get_theme_file_uri()`, `get_template_directory_uri()`
- Fonctions générales PHP : boucles, conditions sur variables définies, tableaux
- Fonctions WordPress générales : `home_url()`, `site_url()`, `get_bloginfo()`

L’utilisation du PHP

Cas d'usage #1 : internationalisation

Vous créez un **thème multilingue** et votre section **hero** dit « **Welcome to WordCamp Nice ♥** »

Comment le traduire ?

- à la mano, dans l’éditeur ?
- dans le template part ?
- **…**

```
<!-- wp:heading {"textAlign":"center"} -->
<h2 class="wp-block-heading has-text-align-center">Welcome to WordCamp Nice ♥</h2>
<!-- /wp:heading -->
```

L’utilisation du PHP

Cas d'usage #1 : internationalisation

```
<!-- wp:heading {"textAlign":"center"} -->
<h2 class="wp-block-heading has-text-align-center"><?php esc_html_e( 'Welcome to WordCamp Nice ❤', 'wc-nice' ); ?></h2>
<!-- /wp:heading -->
```

- ✅ Texte dans le fichier .pot
- ✅ Traduisible via Poedit, Loco Translate, etc.
- ✅ Conforme aux standards WordPress

Cas d'usage #1 : internationalisation

Fonctions d'internationalisation disponibles :

Fonction	Usage	Exemple
`esc_html_e()`	Afficher du texte HTML échappé	`<?php esc_html_e( 'Texte', 'domain' ); ?>`
`esc_html__()`	Retourner du texte HTML échappé	`<?php echo esc_html__( 'Texte', 'domain' ); ?>`
`esc_attr_e()`	Afficher pour attribut HTML	`title="<?php esc_attr_e( 'Titre', 'domain' ); ?>"`
`_e()`	Afficher (sans échappement)	À éviter si possible
`__()`	Retourner (sans échappement)	À éviter si possible

```php
<?php
/**
 * Title: Hero static
 * Slug: wc-nice/hero-static
 * Categories: featured
 */
?>
<!-- wp:cover {"overlayColor":"contrast","isUserOverlayColor":true,"align":"full"} -->
<div class="wp-block-cover alignfull">
    <span aria-hidden="true" class="wp-block-cover__background has-contrast-background-color has-background-dim-100 has-background-dim"></span>
    <div class="wp-block-cover__inner-container">

        <!-- wp:heading {"textAlign":"center"} -->
        <h2 class="wp-block-heading has-text-align-center">Welcome to WordCamp Nice</h2>
        <!-- /wp:heading -->

        <!-- wp:paragraph {"align":"center"} -->
        <p class="has-text-align-center">Join the WordPress community on the French Riviera for a day of talks, workshops, and networking. WordCamp Nice brings together developers, designers, and users in one of the Mediterranean's most vibrant cities.</p>
        <!-- /wp:paragraph -->

        <!-- wp:buttons {"layout":{"type":"flex","justifyContent":"center"}} -->
        <div class="wp-block-buttons">
            <!-- wp:button {"className":"wp-block-button is-style-outline"} -->
            <div class="wp-block-button is-style-outline"><a class="wp-block-button__link wp-element-button">Register</a></div>
            <!-- /wp:button -->
            <!-- wp:button {"className":"wp-block-button is-style-outline"} -->
            <div class="wp-block-button is-style-outline"><a class="wp-block-button__link wp-element-button">Sponsor the event</a></div>
            <!-- /wp:button -->
        </div>
        <!-- /wp:buttons -->

    </div>
</div>
<!-- /wp:cover -->
```

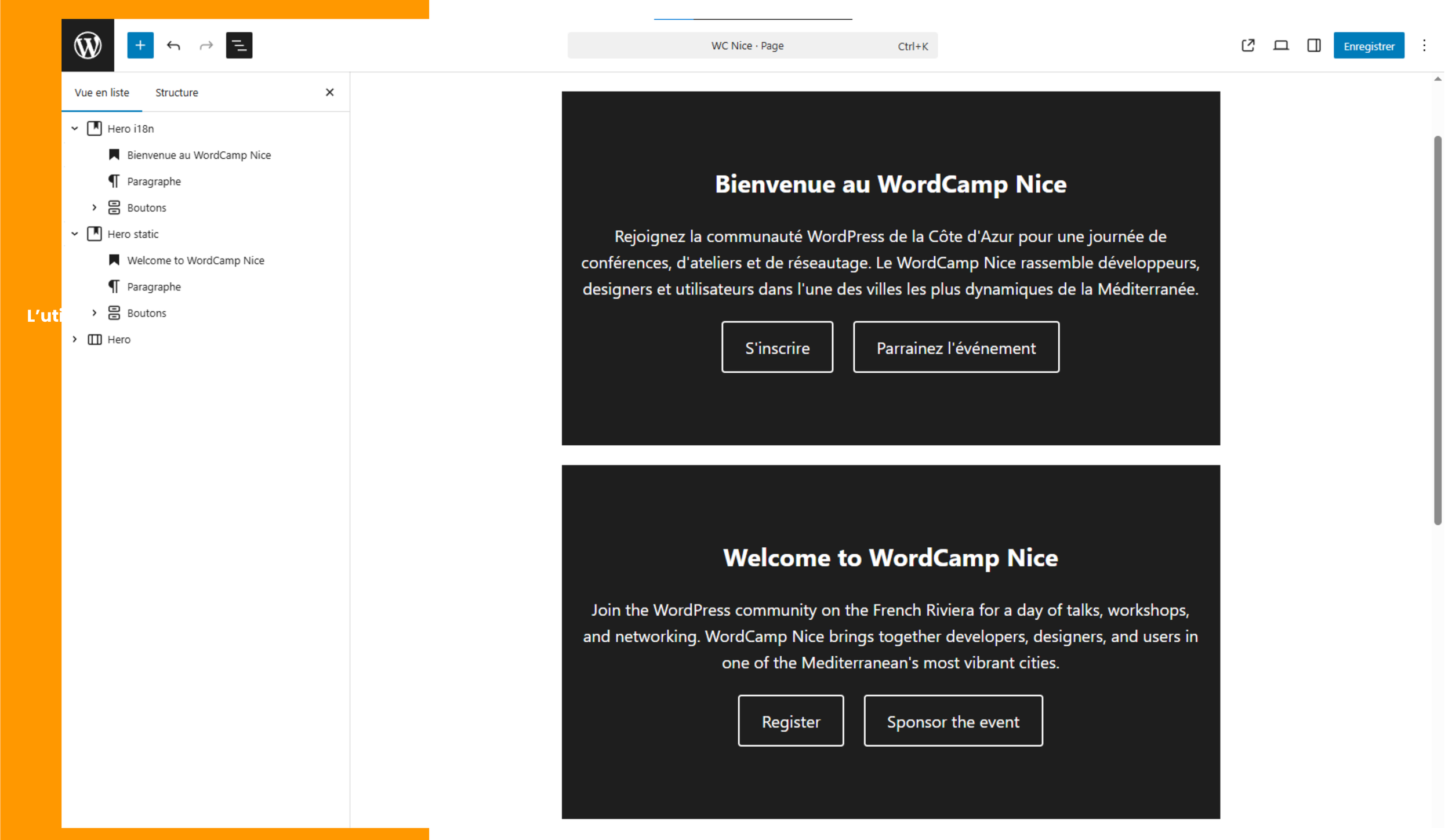
```php
<?php
/**
 * Title: Hero i18n
 * Slug: wc-nice/hero-i18n
 * Categories: featured
 */
?>
<!-- wp:cover {"overlayColor":"contrast","isUserOverlayColor":true,"align":"full"} -->
<div class="wp-block-cover alignfull">
    <span aria-hidden="true" class="wp-block-cover__background has-contrast-background-color has-background-dim-100 has-background-dim"></span>
    <div class="wp-block-cover__inner-container">

        <!-- wp:heading {"textAlign":"center"} -->
        <h2 class="wp-block-heading has-text-align-center"><?php esc_html_e( 'Welcome to WordCamp Nice', 'wc-nice' ); ?></h2>
        <!-- /wp:heading -->

        <!-- wp:paragraph {"align":"center"} -->
        <p class="has-text-align-center"><?php esc_html_e( 'Join the WordPress community on the French Riviera for a day of talks, workshops, and networking.
        WordCamp Nice brings together developers, designers, and users in one of the Mediterranean\'s most vibrant cities.', 'wc-nice' ); ?></p>
        <!-- /wp:paragraph -->

        <!-- wp:buttons {"layout":{"type":"flex","justifyContent":"center"}} -->
        <div class="wp-block-buttons">
            <!-- wp:button {"className":"wp-block-button is-style-outline"} -->
            <div class="wp-block-button is-style-outline"><a class="wp-block-button__link wp-element-button"><?php esc_html_e( 'Register', 'wc-nice' ); ?></a></div>
            <!-- /wp:button -->
            <!-- wp:button {"className":"wp-block-button is-style-outline"} -->
            <div class="wp-block-button is-style-outline"><a class="wp-block-button__link wp-element-button"><?php esc_html_e( 'Sponsor the event', 'wc-nice' ); ?></
            a></div>
            <!-- /wp:button -->
        </div>
        <!-- /wp:buttons -->

    </div>
</div>
<!-- /wp:cover -->
```


WC Nice · Page
Ctrl+K
Enregistrer
Vue en liste
Structure
Hero i18n
Bienvenue au WordCamp Nice
Paragraphe
Boutons
Hero static
Welcome to WordCamp Nice
Paragraphe
Boutons
Hero
Bienvenue au WordCamp Nice
Rejoignez la communauté WordPress de la Côte d'Azur pour une journée de conférences, d'ateliers et de réseautage. Le WordCamp Nice rassemble développeurs, designers et utilisateurs dans l'une des villes les plus dynamiques de la Méditerranée.
S'inscrire
Parrainez l'événement
Welcome to WordCamp Nice
Join the WordPress community on the French Riviera for a day of talks, workshops, and networking. WordCamp Nice brings together developers, designers, and users in one of the Mediterranean's most vibrant cities.
Register
Sponsor the event

L’utilisation du PHP

Cas d'usage #1 : internationalisation

LA BALISE ALT

```
p-content/uploads/2026/02/nicoise.jpg","id":90,"alt":"<?php esc_attr_e( 'Smiling woman from Nice holding a laptop with the WordPress logo on the Promenade des
ze-full" alt="<?php esc_attr_e( 'Smiling woman from Nice holding a laptop with the WordPress logo on the Promenade des Anglais', 'wc-nice' ); ?>" src="https://
div>
```

Cas d'usage #1 : les attributs

LA BALISE ALT

N'oubliez pas les attributs !

Si votre image a un alt, il doit aussi être traduisible.

```
p-content/uploads/2026/02/nicoise.jpg","id":90,"alt":"<?php esc_attr_e( 'Smiling woman from Nice holding a laptop with the WordPress logo on the Promenade des
ze-full" alt="<?php esc_attr_e( 'Smiling woman from Nice holding a laptop with the WordPress logo on the Promenade des Anglais', 'wc-nice' ); ?>" src="https://
div>
```

L’utilisation du PHP

Cas d'usage #1 : les assets

LES IMAGES

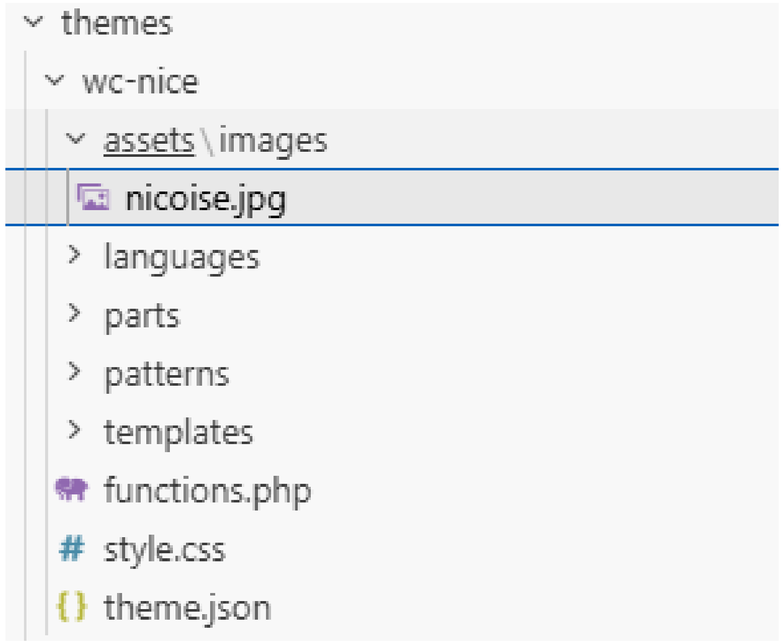
Vous avez une image de background dans votre composition.

En dev, c'est **localhost:8888**.
En prod, c'est **monsite.com**

Cas d'usage #1 : les assets

LES IMAGES

```php
<?php echo esc_url( get_theme_file_uri( 'assets/images/nicoise.jpg' ) ); ?>
```



L’utilisation du PHP

Cas d'usage #1 : les assets

LES IMAGES

⚠ **Important : Double occurrence**

Dans un Cover block avec image, l'URL apparaît **2 fois** :

1. Dans l'attribut JSON du bloc
2. Dans la balise `<img>`

```
<!-- wp:cover {"url":"<?php echo esc_url( get_theme_file_uri( 'assets/images/hero.jpg' ) ); ?
>","dimRatio":50} -->
<div class="wp-block-cover">
    <img class="wp-block-cover__image-background"
         src="<?php echo esc_url( get_theme_file_uri( 'assets/images/hero.jpg' ) ); ?>"
         alt="<?php esc_attr_e( 'Hero background', 'themeslug' ); ?>"
         data-object-fit="cover"/>
    <div class="wp-block-cover__inner-container">
        <!-- contenu -->
    </div>
</div>
<!-- /wp:cover -->
```

L’utilisation du PHP

Cas d'usage #1 : les assets

LES IMAGES

Astuce : Utiliser une variable

Pour éviter la répétition et faciliter la maintenance :

```
<?php
/**
 * Title: Hero avec image
 * Slug: themeslug/hero-image
 * Categories: featured
 */

$hero_image = esc_url( get_theme_file_uri( 'assets/images/hero.jpg' ) );
?>
<!-- wp:cover {"url":"<?php echo $hero_image; ?>","dimRatio":50} -->
<div class="wp-block-cover">
    <img class="wp-block-cover__image-background"
         src="<?php echo $hero_image; ?>"
         alt="<?php esc_attr_e( 'Hero background', 'themeslug' ); ?>"
         data-object-fit="cover"/>
    <!-- ... -->
</div>
<!-- /wp:cover -->
```

L’utilisation du PHP

Cas d'usage #1 : les boucles

APPROCHE DRY (DON'T REPEAT YOURSELF)

```php
$buttons = array(
    'register' => __( 'Register', 'wc-nice' ),
    'sponsor' => __( 'Sponsor the event', 'wc-nice' )
);
```

```
<!-- wp:buttons {"layout":{"type":"flex","justifyContent":"center"}} -->
<div class="wp-block-buttons">
    <?php foreach ( $buttons as $key => $button ) : ?>
        <!-- wp:button {"className":"wp-block-button is-style-outline"} -->
        <div class="wp-block-button is-style-outline"><a class="wp-block-button__link wp-element-button"><?php echo
        esc_html( $button ); ?></a></div>
        <!-- /wp:button -->
    <?php endforeach; ?>
</div>
<!-- /wp:buttons -->
```

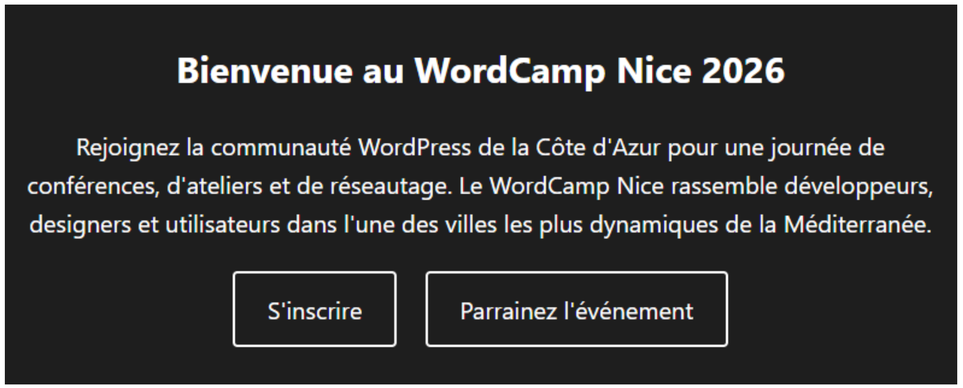
L’utilisation du PHP

Cas d'usage #1 : des variables

```php
<?php
/**
 * Title: Hero static boucle
 * Slug: wc-nice/hero-static-loop
 * Categories: featured
 */

$title = __( 'Welcome to WordCamp Nice', 'wc-nice' );

```




```
<!-- wp:heading {"textAlign":"center"} -->
<h2 class="wp-block-heading has-text-align-center"><?php echo esc_html( sprintf( '%1$s %2$s', $title, date( 'Y' ) ) ); ?></h2>
<!-- /wp:heading -->
```

L’utilisation du PHP

Cas d'usage #1 : des variables

AUTRES USAGES

```php
// Lien vers la page d'accueil
<a href="<?php echo esc_url( home_url( '/' ) ); ?>">

// Lien vers une page spécifique
<a href="<?php echo esc_url( home_url( '/contact' ) ); ?>">

// Nom du site
<?php echo esc_html( get_bloginfo( 'name' ) ); ?>
```

L’utilisation du PHP

Rappels

Performance
Le PHP est exécuté une seule fois à l'enregistrement, pas à chaque affichage. Donc pas d'impact sur les performances front-end.

Sécurité
Échappez TOUJOURS vos outputs. esc_html_e(), esc_url(), esc_attr().

Workflow
Créez votre composition dans l'éditeur d'abord pour avoir le markup correct, puis ajoutez le PHP. Ne codez pas le markup de zéro.

Industrialiser sa pratique

Industrialiser sa pratique

Le problème du workflow actuel

LE CYCLE INFERNAL

1. Créer pattern dans l'éditeur WordPress
   ↓
2. Copier le markup (Options > Copier tous les blocs)
   ↓
3. Coller dans /patterns/pattern.php
   ↓
4. Ajouter le header du fichier
   ↓
5. Client demande une modif
   ↓
6. Modifier dans l'éditeur pour prévisualiser
   ↓
7. Re-copier tout le markup
   ↓
8. Écraser le fichier...
   ↓
9. RÉPÉTER 50 FOIS 😱

Industrialiser sa pratique

Organiser son dossier /patterns

STRUCTURE RECOMMANDÉE

```
/patterns/
├── hero-main.php
├── hero-simple.php
├── hero-video.php
├── cta-newsletter.php
├── cta-contact.php
├── team-grid.php
├── team-carousel.php
├── footer-minimal.php
├── footer-complete.php
└── hidden-query-grid.php
```

Industrialiser sa pratique

Organiser son dossier /patterns

CONVENTIONS DE NOMMAGE

Préfixe	Usage	Exemples
`hero-`	Sections d'en-tête/bannières	`hero-main.php`, `hero-video.php`
`cta-`	Call-to-action	`cta-newsletter.php`, `cta-contact.php`
`team-`	Équipe	`team-grid.php`, `team-list.php`
`footer-`	Pieds de page	`footer-minimal.php`, `footer-complete.php`
`header-`	En-têtes de site	`header-centered.php`, `header-split.php`
`query-`	Boucles Query Loop	`query-grid.php`, `query-list.php`
`hidden-`	Patterns non visibles dans l'inserter	`hidden-template-home.php`

Industrialiser sa pratique

Organiser son dossier /patterns

CONVENTIONS DE NOMMAGE

💡 **Astuce : préfixe pour template patterns**

Je préfixe mes patterns destinés uniquement aux templates (non visibles dans l'inserter) avec `hidden-`. Ça les regroupe et je sais immédiatement leur usage.

Avantages d'une bonne organisation :

- Recherche rapide dans l'IDE
- Cohérence entre projets
- Onboarding facilitée pour les nouveaux développeurs
- Maintenance simplifiée

Industrialiser sa pratique

Headers de fichiers

LE HEADER MINIMAL

```
<?php
/**
 * Title: Hero
 * Slug: wc-nice/hero
 * Categories: featured
 */
?>
```

Industrialiser sa pratique

Headers de fichiers

LE HEADER COMPLET

```
<?php
/**
 * Title: Hero Principal
 * Slug: wc-nice/hero-main
 * Categories: featured, banner
 * Description: Section hero avec image de fond, titre principal et boutons d'action. Idéal pour page d'accueil.
 * Keywords: accueil, bannière, principal, home, landing
 * Viewport Width: 1376
 * Block Types: core/post-content
 * Post Types: page
 * Inserter: true
 */
?>
```

Industrialiser sa pratique

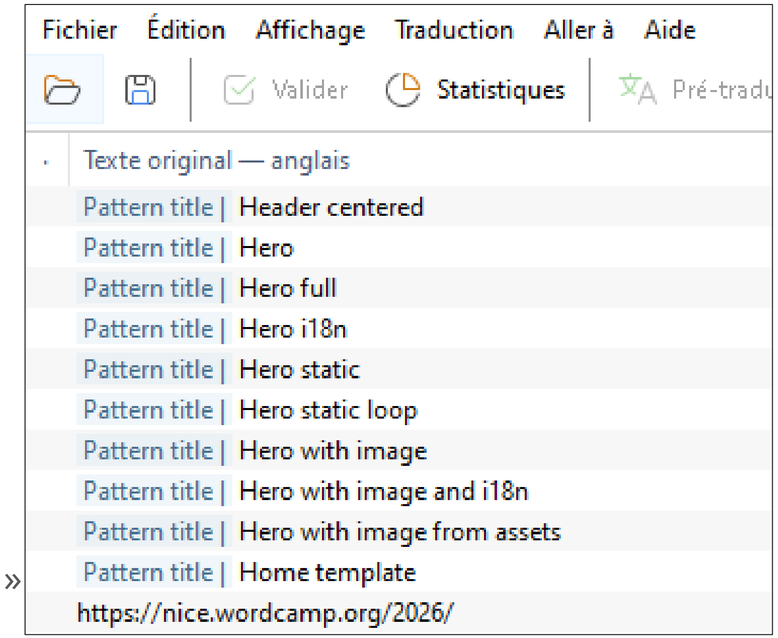
Headers de fichiers

DÉTAIL DE CHAQUE CHAMP

Title (obligatoire)

Title: Hero Principal

- Nom affiché dans l'interface
- Peut être traduit (mais pas automatiquement)
- Soyez descriptif : « Hero Principal » plutôt que « Hero »



Depuis le dossier du thème :

- **wp i18n make-pot . languages/wc-nice.pot --domain=wc-nice**

Industrialiser sa pratique

Headers de fichiers

DÉTAIL DE CHAQUE CHAMP

Slug (obligatoire)

Slug: themeslug/hero-main

- Identifiant unique : **namespace/nom**
- Namespace = slug de votre thème
- Utilisé pour l'inclusion via <!-- wp:pattern {"slug":"..."} /-->

Industrialiser sa pratique

Headers de fichiers

DÉTAIL DE CHAQUE CHAMP

Categories (recommandé)

Categories: featured, banner, themeslug-custom

- Catégories où le pattern apparaît
- Peut inclure des catégories custom
- Séparées par des virgules

Catégories WordPress par défaut :
featured, about, audio, banner, buttons, call-to-action, columns, contact, footer, gallery, header, media, portfolio, posts, services, team, testimonials, text, video

Industrialiser sa pratique

Headers de fichiers

DÉTAIL DE CHAQUE CHAMP

Description

Description: Section hero avec image de fond, titre principal et boutons d'action.

- Uniquement visible pour les lecteurs d'écran
- Améliore l'accessibilité
- Bonne pratique souvent négligée

Industrialiser sa pratique

Headers de fichiers

DÉTAIL DE CHAQUE CHAMP

Keywords

Keywords: accueil, bannière, principal, home, landing, hero

Game-changer pour l'UX

- Améliore la recherche dans l'inserter
- Mots-clés séparés par des virgules
- Incluez des synonymes et termes de recherche probables

💡 **Astuce :**
Vos clients cherchent « bannière », pas « hero ». **Mettez les deux en keywords**.

Industrialiser sa pratique

Headers de fichiers

DÉTAIL DE CHAQUE CHAMP

Viewport Width

Viewport Width: 1376

- Largeur de la preview dans l'inserter
- En pixels
- Permet un aperçu réaliste
- Valeurs courantes : 1376, 1280, 1024

Industrialiser sa pratique

Headers de fichiers

DÉTAIL DE CHAQUE CHAMP

Block Types

Block Types: core/post-content, core/cover

- Associe le pattern à des types de blocs spécifiques
- core/post-content = Starter page pattern
- core/template-part/header = Pattern de remplacement header
- core/query = Pattern Query Loop

Cas d'usage :

- Pattern proposé lors de l'insertion d'un bloc spécifique
- Starter patterns pour pages
- Variantes de header / footer
- ...

Industrialiser sa pratique

Headers de fichiers

DÉTAIL DE CHAQUE CHAMP

Block Types: core/post-content

👉 WordPress le propose comme **Starter Pattern de page**.

Cas typique :

- A la création d'une nouvelle page vide
- WordPress propose des mises en page prêtes à l'emploi

C'est le mécanisme des « Starter patterns ».

Industrialiser sa pratique

Headers de fichiers

DÉTAIL DE CHAQUE CHAMP

core/template-part/header

👉 Le pattern est proposé quand nous modifions un header.

Cas d’usage :

- Remplacement complet d’un header
- Bibliothèque de variations de header

Très utile en thème FSE

Industrialiser sa pratique

Headers de fichiers

DÉTAIL DE CHAQUE CHAMP

Block Types: core/query

👉 Le pattern apparaît comme alternative quand nous insérons un bloc Query.

Donc :

- On ajoute une Query Loop
- WordPress peut proposer notre layout custom (grid, masonry, list…)

Industrialiser sa pratique

Headers de fichiers

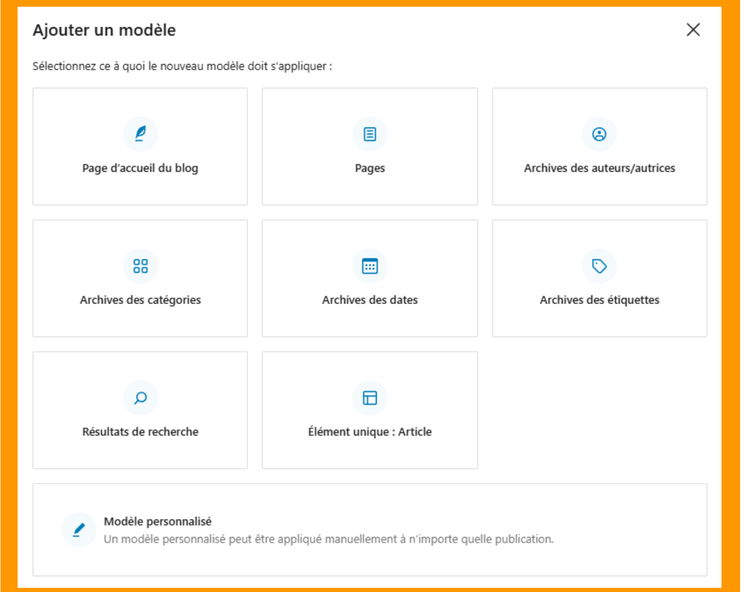
DÉTAIL DE CHAQUE CHAMP

Post Types

Post Types: page, post, portfolio

- Limite le pattern à certains types de contenu
- Par défaut : disponible partout
- Utile pour patterns spécifiques

Industrialiser sa pratique



Headers de fichiers

DÉTAIL DE CHAQUE CHAMP

Template Types (pour starter template patterns)

Template Types: front-page, home, single

- Pattern proposé lors de la création d'un template
- Types supportés :
  index, home, front-page, singular, single, page, archive, author, category, taxonomy, date, tag, attachment, search, privacy-policy, 404

Industrialiser sa pratique

Headers de fichiers

DÉTAIL DE CHAQUE CHAMP

Inserter

Inserter: false

- true (défaut) : visible dans l'inserter
- false : masqué de l'inserter
- Utile pour patterns utilisés uniquement dans les templates

Si **false**, alors il est masqué mais :

- Il peut être appelé dans un **template**
- Il peut être utilisé via **<!-- wp:pattern -->**
- Mais il n’est pas visible dans le bouton “**+**”

Industrialiser sa pratique

Headers de fichiers

DÉTAIL DE CHAQUE CHAMP

⚠ **Attention syntaxe :**

- Pas de virgule après la dernière ligne
- Respecter la casse exacte des clés
- Pas d'espaces superflus

Industrialiser sa pratique

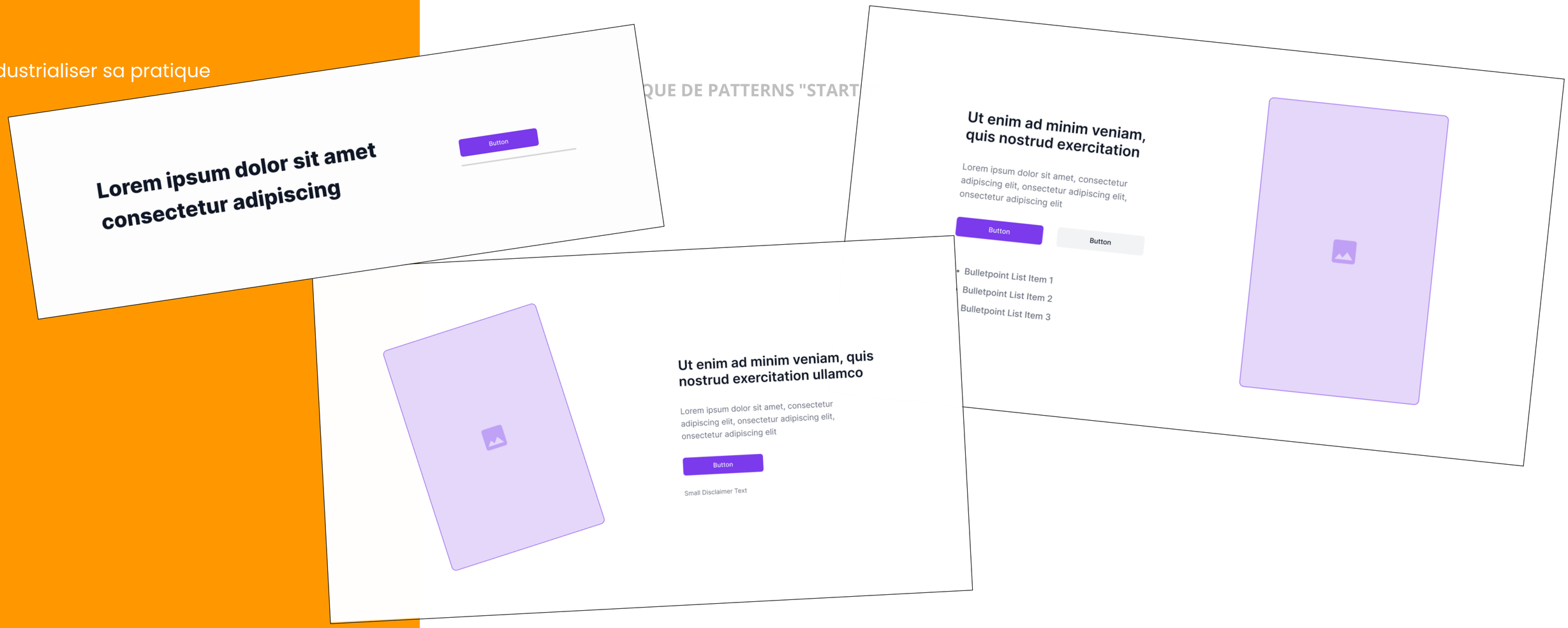
Réutilisation entre projets

CRÉER UNE BIBLIOTHÈQUE DE PATTERNS "STARTER

```
/starter-patterns/
├── heroes/
│   ├── hero-centered.php
│   ├── hero-split.php
│   └── hero-video.php
├── ctas/
│   ├── cta-newsletter.php
│   └── cta-contact.php
├── footers/
│   ├── footer-minimal.php
│   └── footer-complete.php
└── README.md
```

Industrialiser sa pratique

Réutilisation entre projets



Industrialiser sa pratique

Réutilisation entre projets

CRÉER UNE BIBLIOTHÈQUE DE PATTERNS "STARTER

Workflow

- **Développer** des patterns génériques (pas de couleurs/typos en dur)
- **Versionner** dans un repo Git séparé
- **Copier** dans chaque nouveau projet
- **Adapter** uniquement via theme.json (couleurs, typos, espacements)

Industrialiser sa pratique

Réutilisation entre projets

CRÉER UNE BIBLIOTHÈQUE DE PATTERNS "STARTER

Pattern générique vs spécifique

💡 Astuce : utiliser les presets

Concevez vos patterns avec des presets (primary, secondary, contrast, base) plutôt que des valeurs en dur. Le theme.json du projet définira les vraies couleurs.

```
// ❌ Spécifique (couleur en dur)
<!-- wp:group {"backgroundColor":"#FF5733"} -->

// ✅ Générique (preset)
<!-- wp:group {"backgroundColor":"primary"} -->
```

Industrialiser sa pratique

Réutilisation entre projets

CRÉER UNE BIBLIOTHÈQUE DE PATTERNS "STARTER

Checklist pour un pattern réutilisable

- Couleurs via presets, pas de hex en dur
- Espacements via presets (spacing.40, spacing.50, etc.)
- Textes internationalisés avec textdomain générique (**themeslug à remplacer**)
- Pas d'URLs en dur
- Documentation claire dans le fichier

Contrôler l’expérience d’édition

ADAPTER L’ÉDITEUR WORDPRESS POUR QU’IL SOIT FACILE À UTILISER

Contrôler l’expérience d’édition

Désactiver les patterns Core

CONTRÔLE PROGRAMMATIQUE

```php
// Dans functions.php
add_action( 'after_setup_theme', 'themeslug_remove_core_patterns' );

function themeslug_remove_core_patterns() {
    remove_theme_support( 'core-block-patterns' );
}
```

Contrôler l’expérience d’édition

Désactiver les patterns distants (Pattern Directory)

CONTRÔLE PROGRAMMATIQUE

```php
// Dans functions.php
add_filter( 'should_load_remote_block_patterns', '__return_false' );
```

Contrôler l’expérience d’édition

Créer une catégorie personnalisée

CONTRÔLE PROGRAMMATIQUE

```php
add_action( 'init', 'themeslug_register_pattern_categories' );

function themeslug_register_pattern_categories() {
    register_block_pattern_category( 'themeslug-custom', array(
        'label'       => __( 'Mon Thème', 'themeslug' ),
        'description' => __( 'Patterns personnalisés pour ce thème', 'themeslug' ),
    ) );
}
```

Conditionner l’affichage

Contrôler l’expérience d’édition

ENREGISTREMENT CONDITIONNEL

```php
add_action( 'init', 'themeslug_conditional_patterns', 999 );

function themeslug_conditional_patterns() {
    // Désactiver un pattern si WooCommerce n'est pas actif
    if ( ! class_exists( 'WooCommerce' ) ) {
        unregister_block_pattern( 'themeslug/product-grid' );
    }

    // Ou enregistrer conditionnellement
    if ( class_exists( 'WooCommerce' ) ) {
        register_block_pattern( 'themeslug/product-grid', array(
            'title'      => __( 'Grille Produits', 'themeslug' ),
            'categories' => array( 'featured' ),
            'source'     => 'theme',
            'content'    => '<!-- markup -->'
        ) );
    }
}
```

Contrôler l’expérience d’édition

Le vrai pouvoir : block Locking API

Vos clients ne sont pas designers. Ils ne devraient pas avoir à l'être !

Les problèmes récurrents :

- Client qui supprime accidentellement un bloc crucial
- Design cassé par des ajouts non prévus
- Mise en page déstructurée
- « J'ai tout cassé, vous pouvez remettre comme avant ? » 😱

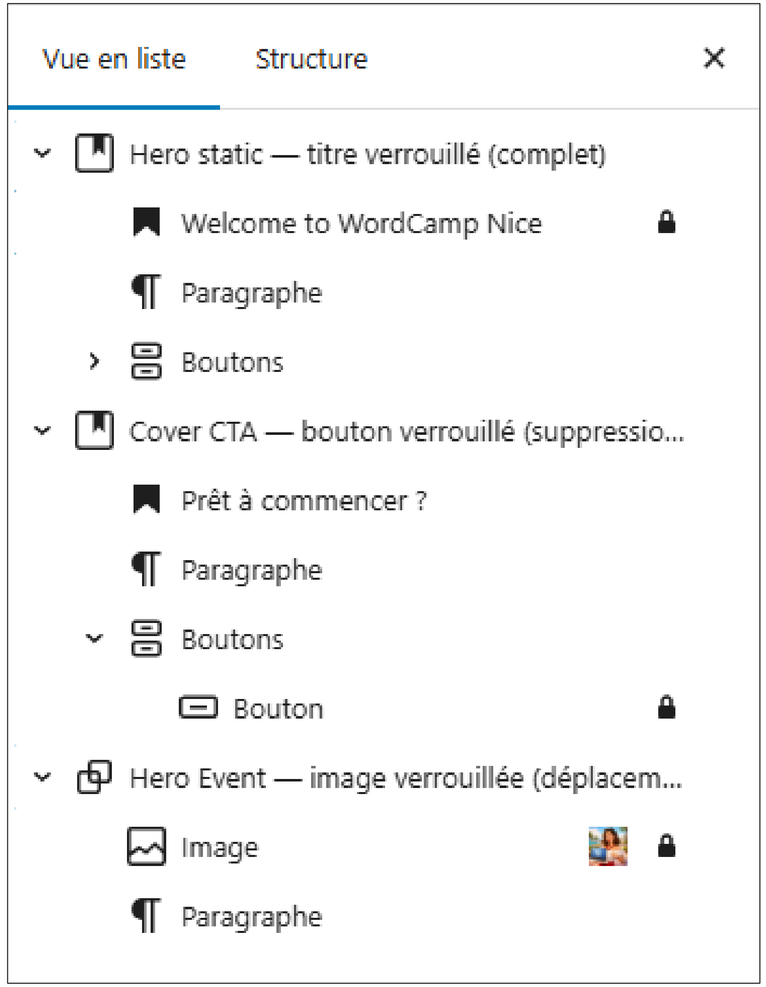
Contrôler l’expérience d’édition

Ne pas mettre le **oaï**

La solution : Block Locking API

Contrôler ce que l'utilisateur peut faire avec les blocs :

- Empêcher le déplacement
- Empêcher la suppression
- Empêcher l'ajout de nouveaux blocs
- Limiter l'édition au contenu uniquement



Contrôler l’expérience d’édition

Le vrai pouvoir : block Locking API

Deux niveaux de verrouillage

- **Block Level** - Verrouiller un bloc individuel
- **Template Level** - Verrouiller un groupe de blocs imbriqués

Contrôler l’expérience d’édition

Block Level Locking

Syntaxe de base

{"lock":{"move":true,"remove":true}}

- ***move: true*** = déplacement verrouillé
- ***remove: true*** = suppression verrouillée

Contrôler l’expérience d’édition

Block Level Locking

Application dans le markup

```
<!-- wp:image {"sizeSlug":"full","lock":{"move":true,"remove":false}} -->
<figure class="wp-block-image size-full">
    <img src="..." alt="..."/>
</figure>
<!-- /wp:image -->
```

Contrôler l’expérience d’édition

Block Level Locking

EXEMPLE 1 : EMPÊCHER LE DÉPLACEMENT

Cas d'usage : l'image hero doit toujours rester en haut du pattern

- L'image ne peut pas être déplacée
- L'image peut être supprimée (si c’est vraiment voulu)
- Le contenu de l'image (source, alt) reste éditable

Contrôler l’expérience d’édition

Block Level Locking

```
<?php
/**
 * Title: Event
 * Slug: themeslug/event
 * Categories: banner
 */
?>
<!-- wp:group {"align":"wide","backgroundColor":"contrast","textColor":"base-2"} -->
<div class="wp-block-group alignwide has-base-2-color has-contrast-background-color has-text-color has-background">

    <!-- wp:image {"sizeSlug":"full","lock":{"move":true,"remove":false}} -->
    <figure class="wp-block-image size-full">
        <img src="https://example.com/event.jpg" alt="Event"/>
    </figure>
    <!-- /wp:image -->

    <!-- wp:paragraph -->
    <p><?php esc_html_e( 'Event details here', 'themeslug' ); ?></p>
    <!-- /wp:paragraph -->

</div>
<!-- /wp:group -->
```

Contrôler l’expérience d’édition

Block Level Locking

EXEMPLE 2 : EMPÊCHER LA SUPPRESSION

Cas d'usage : Le bouton CTA est obligatoire dans le design.

- Le bouton ne peut pas être supprimé
- Le bouton peut être déplacé
- Le texte et le lien restent éditables

```
<!-- wp:button {"lock":{"move":false,"remove":true}} -->
<div class="wp-block-button">
    <a class="wp-block-button__link wp-element-button"><?php esc_html_e( 'Contact Us', 'themeslug' ); ?></a>
</div>
<!-- /wp:button -->
```

Contrôler l’expérience d’édition

Block Level Locking

EXEMPLE 3 : VERROUILLAGE COMPLET

Cas d'usage : Ce bloc ne doit absolument pas bouger ni disparaître.

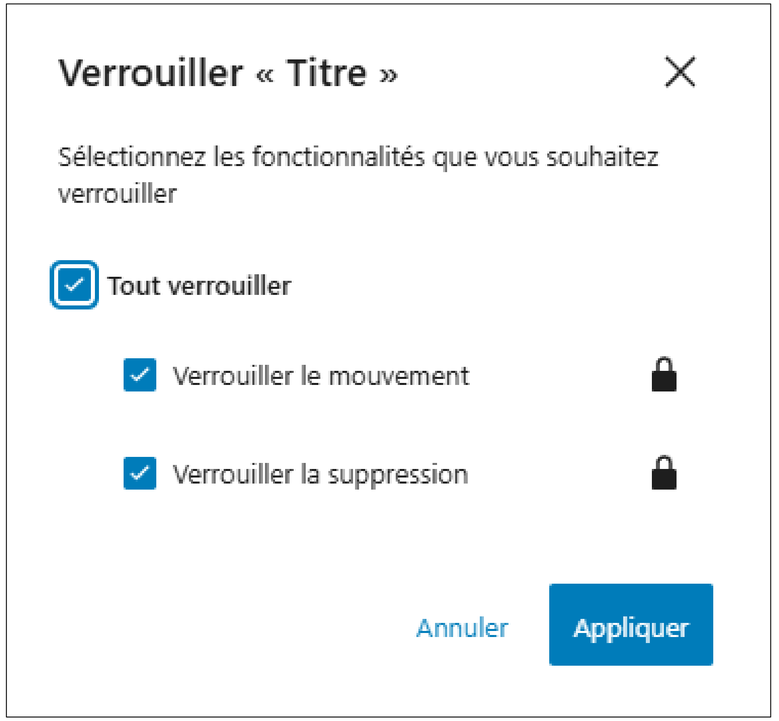
- Impossible de déplacer
- Impossible de supprimer
- Texte toujours éditable

```
<!-- wp:heading {"lock":{"move":true,"remove":true}} -->
<h2><?php esc_html_e( 'Important Title', 'themeslug' ); ?></h2>
<!-- /wp:heading -->
```

Contrôler l’expérience d’édition

Block Level Locking

⚠ **Important : déverrouillage par l'utilisateur**



Par défaut, les utilisateurs peuvent déverrouiller les blocs via l'interface.

Contrôler l’expérience d’édition

Template Level Locking

Le template locking s'applique à un **bloc conteneur** (Group, Cover, Columns, Column, Navigation) et affecte tous ses blocs imbriqués

⚠ **Ne fonctionne PAS sur :** Paragraph, Heading, Image, Button (blocs non-conteneurs)

Syntaxe de base

{"templateLock":"all|insert|contentOnly"}

Contrôler l’expérience d’édition

Template Level Locking

OPTION 1 : `TEMPLATELOCK: "ALL"`

Effet : applique le verrouillage move/remove à TOUS les blocs imbriqués.

Résultat

❌ Aucun bloc ne peut être déplacé
❌ Aucun bloc ne peut être supprimé
✅ Le contenu (texte, images…) reste éditable
✅ Les settings des blocs (couleurs, styles…) restent accessibles

```
<!-- wp:group {"templateLock":"all","lock":{"move":true,"remove":true}} -->
<div class="wp-block-group">
    <!-- Tous les blocs ici sont verrouillés -->
</div>
<!-- /wp:group -->
```

```
<!-- wp:group {"templateLock":"insert","align":"wide"} -->
<div class="wp-block-group alignwide">

    <!-- wp:columns -->
    <div class="wp-block-columns">

        <!-- wp:column -->
        <div class="wp-block-column">
            <!-- wp:heading {"level":3} -->
            <h3><?php esc_html_e( 'Feature 1', 'themeslug' ); ?></h3>
            <!-- /wp:heading -->
        </div>
        <!-- /wp:column -->

        <!-- wp:column -->
        <div class="wp-block-column">
            <!-- wp:heading {"level":3} -->
            <h3><?php esc_html_e( 'Feature 2', 'themeslug' ); ?></h3>
            <!-- /wp:heading -->
        </div>
        <!-- /wp:column -->

        <!-- wp:column -->
        <div class="wp-block-column">
            <!-- wp:heading {"level":3} -->
            <h3><?php esc_html_e( 'Feature 3', 'themeslug' ); ?></h3>
            <!-- /wp:heading -->
        </div>
        <!-- /wp:column -->

    </div>
    <!-- /wp:columns -->

</div>
<!-- /wp:group -->
```

te Level Locking

OPTION 2 : `TEMPLATELOCK: "INSERT"`

Effet : impossible d'ajouter de nouveaux blocs dans le conteneur.

Résultat

❌ Impossible d'ajouter une 4ème colonne ou tout autre bloc

✅ Blocs existants éditables (contenu et settings)

✅ Blocs existants déplaçables/supprimables

Contrôler l’expérience d’édition

Template Level Locking

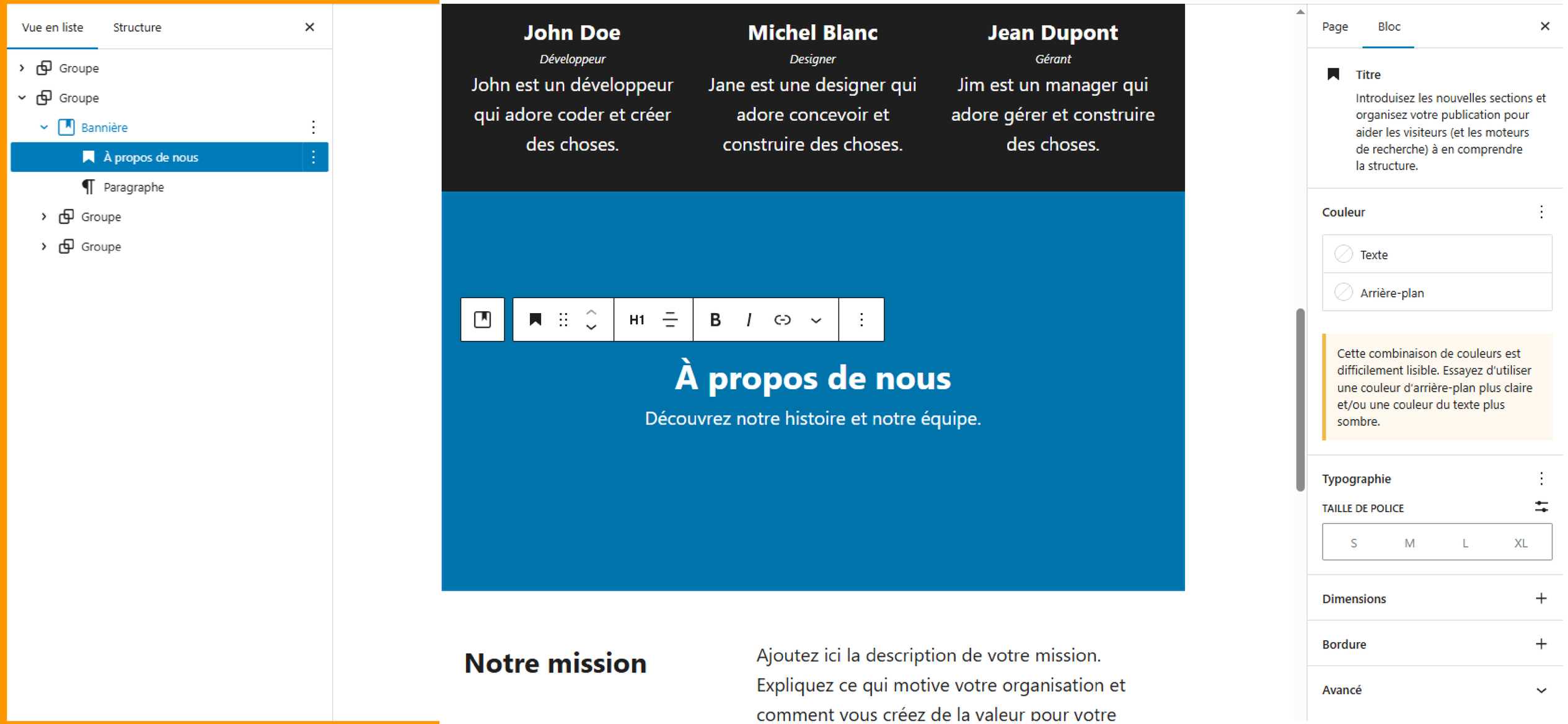
OPTION 3 : `TEMPLATELOCK: "CONTENTONLY"`

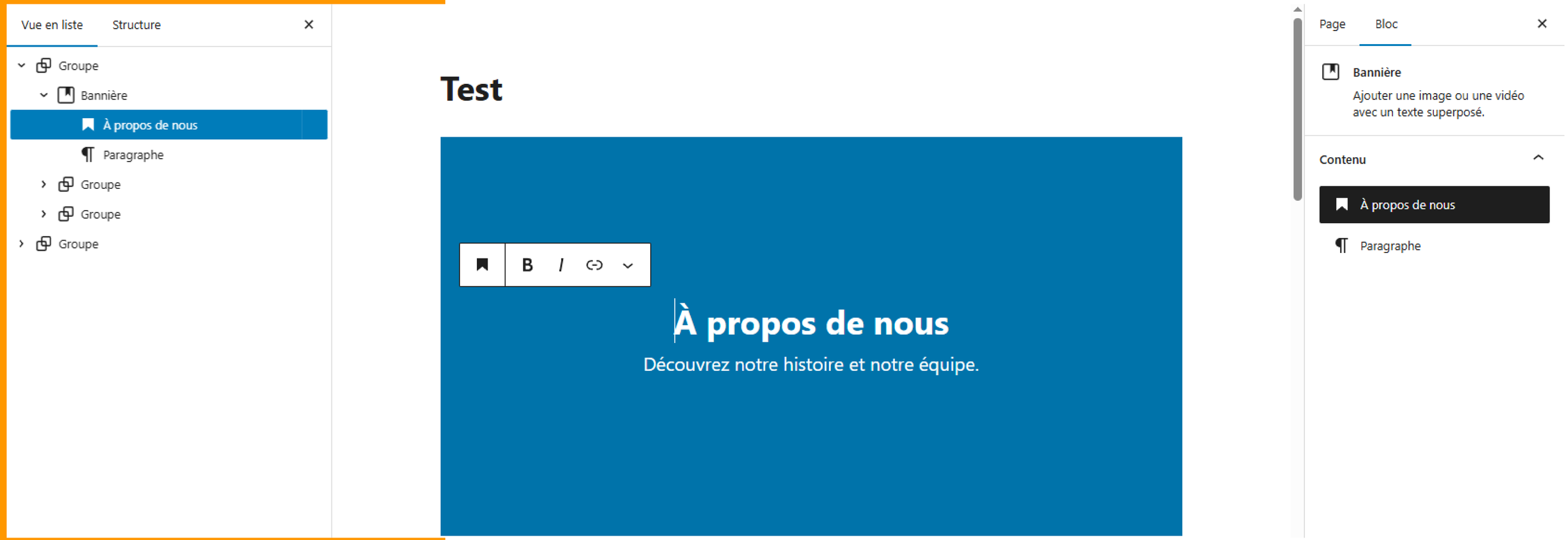
Effet : édition limitée au contenu uniquement (texte et médias).
Aucun accès aux settings de design

```
<!-- wp:group {"templateLock":"contentOnly"} -->
<div class="wp-block-group">
    <!-- Seul le contenu est éditable -->
</div>
<!-- /wp:group -->
```

Ce qui change pour l'utilisateur

Sans contentOnly	Avec contentOnly
Sidebar avec tous les settings	Sidebar avec panel "Content" uniquement
Accès aux couleurs, typographies, espacements	Pas d'accès aux settings de design
Toolbar complète	Toolbar simplifiée
Possibilité de tout casser	Interface protégée


Vue en liste
Structure
Groupe
Groupe
Bannière
À propos de nous
Paragraphe
Groupe
Groupe
John Doe
Développeur
John est un développeur qui adore coder et créer des choses.
Michel Blanc
Designer
Jane est une designer qui adore concevoir et construire des choses.
Jean Dupont
Gérant
Jim est un manager qui adore gérer et construire des choses.
H1
À propos de nous
Découvrez notre histoire et notre équipe.
Notre mission
Ajoutez ici la description de votre mission. Expliquez ce qui motive votre organisation et comment vous créez de la valeur pour votre
Page
Bloc
Titre
Introduisez les nouvelles sections et organisez votre publication pour aider les visiteurs (et les moteurs de recherche) à en comprendre la structure.
Couleur
Texte
Arrière-plan
Cette combinaison de couleurs est difficilement lisible. Essayez d'utiliser une couleur d'arrière-plan plus claire et/ou une couleur du texte plus sombre.
Typographie
TAILLE DE POLICE
S
M
L
XL
Dimensions
Bordure
Avancé


Vue en liste
Structure
Groupe
Bannière
À propos de nous
Paragraphe
Groupe
Groupe
Groupe
Test
À propos de nous
Découvrez notre histoire et notre équipe.
Page
Bloc
Bannière
Ajouter une image ou une vidéo avec un texte superposé.
Contenu
À propos de nous
Paragraphe

Contrôler l’expérience d’édition

Contrôle des permissions

Par défaut : tous les utilisateurs peuvent déverrouiller les blocs.

Pour restreindre

```php
add_filter( 'block_editor_settings_all', 'themeslug_restrict_locking' );

function themeslug_restrict_locking( $settings ) {
    // Seuls les admins peuvent déverrouiller
    $settings['canLockBlocks'] = current_user_can( 'manage_options' );

    return $settings;
}
```

Contrôler l’expérience d’édition

Contrôle des permissions

Ou plus granulaire

```
add_filter( 'block_editor_settings_all', 'themeslug_restrict_locking' );

function themeslug_restrict_locking( $settings ) {
    // Admins et éditeurs peuvent déverrouiller
    if ( current_user_can( 'edit_others_posts' ) ) {
        $settings['canLockBlocks'] = true;
    } else {
        $settings['canLockBlocks'] = false;
    }

    return $settings;
}
```

Contrôler l’expérience d’édition

Contrôle des permissions

Résultat

- Le bouton "Lock" n'apparaît plus pour les utilisateurs non autorisés
- Les verrous sont vraiment verrouillés

« Cette restriction est idéale pour les projets clients où vous voulez un contrôle total »

Checklist pour vos prochains projets

Checklist pour vos prochains projets

PHP et dynamisme

- ❑ Utiliser esc_html_e() pour tous les textes visibles
- ❑ Utiliser get_theme_file_uri() pour les assets du thème
- ❑ Appliquer le principe DRY avec des boucles foreach()
- ❑ Échapper TOUS les outputs (esc_html, esc_url, esc_attr)
- ❑ Respecter le contexte d'exécution (pas de is_page(), the_title(), etc.)

Checklist pour vos prochains projets

Organisation et industrialisation

❑ Structurer le dossier /patterns avec des conventions de nommage
❑ Exploiter TOUS les headers de fichiers :

- Title, Slug, Categories (obligatoires)
- Keywords (améliore la recherche)
- Viewport Width (meilleure preview)
- Block Types, Post Types, Template Types (cas spécifiques)

Checklist pour vos prochains projets

Contrôler l’expérience d’édition

- ❑ Désactiver les patterns Core si non désirés
- ❑ Désactiver les patterns distants si contrôle total souhaité
- ❑ Créer des catégories personnalisées pour organiser
- ❑ Identifier les patterns nécessitant un verrouillage
- ❑ Choisir le bon niveau de lock :
  - Block level (move/remove) pour blocs individuels
  - Template level insert pour empêcher les ajouts
  - Template level contentOnly pour édition contenu uniquement
- ❑ Implémenter des starter page patterns pour l'onboarding
- ❑ Créer des template patterns pour les templates récurrents
- ❑ Restreindre les permissions si projet client critique

Thierry **Pigot**

Architecte WP

Direction technique

SEO

contact@wearewp.pro
06 01 95 63 81

https://github.com/thierrypigot/wc-nice

WP.ASSISTANCE Support & Maintenance WordPress

weare[wp]

weare[wp]

À la fois **Agence WordPress** & **Agence de communication digitale**,
nous mettons au profit de nos clients nos 20 ans d'expertise
dans le domaine du web et des stratégies digitales.

E-mail : contact@wearewp.pro

Tél. : 06 01 95 63 81