Arrêtez d'écrire du PHP*, passez au Full Site Editing

ADIEU THE LOOP ! 🎊

```php
<?php if ( have_posts() ) : while ( have_posts() ) : the_post(); ?>
...
<?php endwhile; else : ?>
 <p><?php esc_html_e( 'Sorry, no posts matched your criteria.' ); ?></p>
<?php endif; ?>
```

Télécharger la présentation en format PDF

*dans vos templates

Faramaz Patrick

Développeur depuis plus de 15 ans
Indépendant depuis 2016 (FR, CH)

ReactJS - VueJS - Next.js - Nuxt.js - WordPress - Craft CMS

Co-présentateur du podcast Double Slash

Podcast bimensuel sur le développement web moderne

https://double-slash.dev/

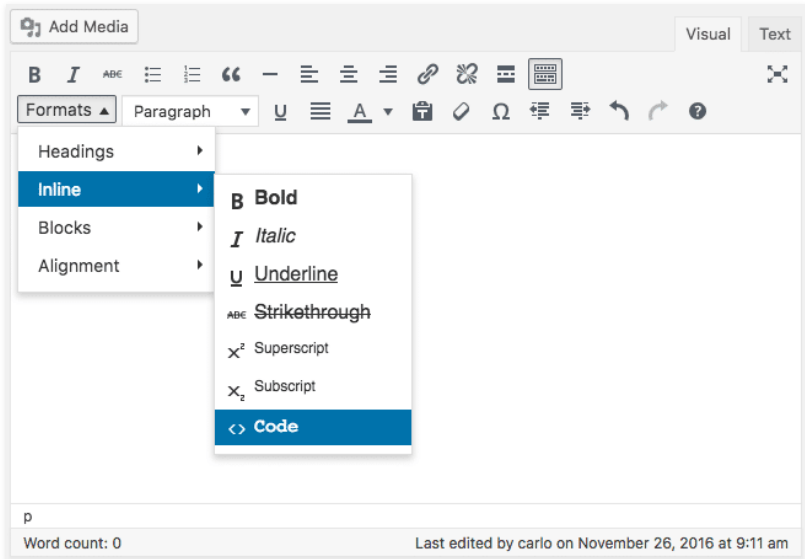
But de la présentation

Expliquer les bases du Full Site Editing pour vous donner envie de tester et d'approfondir le sujet.

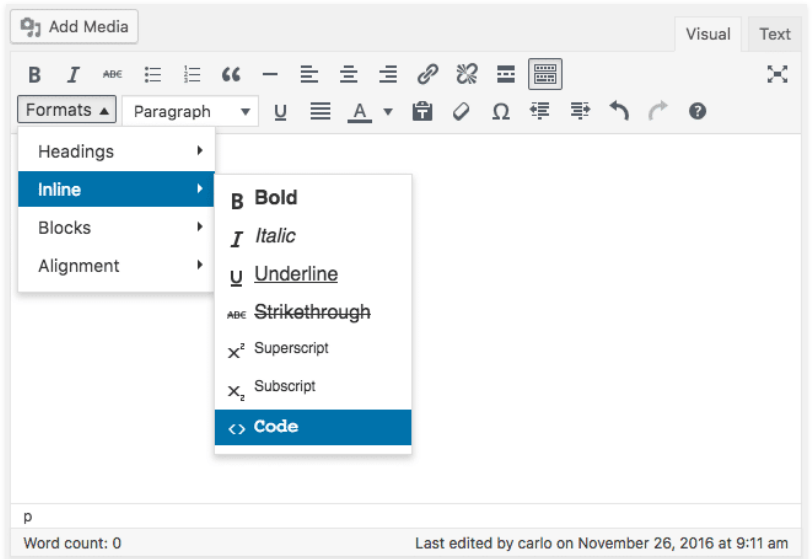
Et pourquoi pas sortir un premier site dans quelques mois en mode FSE 🚀

Rapide Historique

Avant l’arrivée de Gutenberg, WordPress ne proposait que TinyMCE en tant qu’éditeur de contenu.



Avant l'arrivée de Gutenberg, WordPress ne proposait que TinyMCE en tant qu'éditeur de contenu.



En 2005, l'éditeur TinyMCE a été intégré à Wordpress 2.0 et continue d'être utilisé dans l'éditeur WordPress Classic.

Trop limité pour les développeurs.euses et les utilisateurs.trices

C'est donc à cause de cette faiblesse que l'on a vu l'apparition de différentes alternatives :

- shortcodes (mauvaise UX)
- ACF avec les flexibles contents (mauvaise UX)
- Les pages builders tels que Divi, Elementor, etc. (...)

Ces derniers ont pris une grosse part de marché en attendant que WordPress sorte enfin une vraie solution.

L’idée du changement

2016

L’idée d’un nouvel éditeur visuel plus avancé a été initiée par Matt Mullenweg (créateur de WordPress) lors du WordCamp US **2016**.

2017

Le projet Gutenberg a donc commencé en 2017, ce qui a conduit à la création d’une équipe de développement dédiée à ce projet. Cette équipe a travaillé sur Gutenberg jusqu’à sa sortie en **2018**

2018

Gutenberg est donc un éditeur de contenu pour WordPress. Il a été intégré pour la première fois dans la version 5.0 du CMS en décembre **2018**. Soit **5 ans** déjà !!!

Une sortie un peu chaotique !

PAS PRÊT ET PAS UN BUILDER !

À sa sortie, Gutenberg n'était qu'un éditeur de contenu et son interface n'était pas du tout la même qu'aujourd'hui, et encore moins un équivalent d'un constructeur de pages tel qu'Elementor.

Dans un premier temps, il y a eu une forte opposition à ce changement. Une grande partie de la communauté était contre l'arrivée de Gutenberg. 🛑

Nous avons vu la sortie d'un fork de WordPress sans Gutenberg : "ClassicPress". 🏴‍☠️

L'un des plugins les plus populaires encore aujourd'hui est "Disable Gutenberg" avec plus de **700 000** installations. 🤯

L’arrivé du Full site editing

Le FSE, ou "éditeur de site", permet aux utilisateurs.rices de personnaliser le site entier, y compris les en-têtes, les pieds de page, la navigation et les modèles de page, etc... directement depuis l’éditeur de site.

WORDPRESS 5.8 - JUILLET 2021

Il a été introduit pour la première fois dans WordPress 5.8 en juillet 2021 et fait partie de la phase 2 de la feuille de route.

Pourquoi utiliser Gutenberg et le FSE ?

1 - Parce que c’est natif et dans le core de WordPress

Pour utiliser Gutenberg, vous n’avez pas besoin d’installer un plugin ou de payer une licence. C’est disponible directement dans une nouvelle installation de WordPress.

Le développement de l’éditeur Gutenberg peut-être testé et suivi via le plugin. Le plugin embarque les évolutions de l’éditeur avant d’être mergé dans le core de Wordpress à chaque sortie d’une nouvelle version.

Comme pour le CMS, Gutenberg est totalement open-source.

2 - Une interface simple (presque) et rapide

L'interface est totalement intégrée à l'administration WordPress et les fonctionnalités restent en majorité intuitives.

Depuis les premières versions, l'éditeur Gutenberg a été largement optimisé et il fonctionne assez rapidement. Quand on veut éditer un contenu, l'éditeur se charge sans latence.

Votre admin reste rapide et c'est très important pour une personne qui passe beaucoup de temps dans l'admin.

3 - Le code HTML qui en sort est assez concis

Le HTML généré par Gutenberg est plutôt propre et la majorité du style des éléments est basée sur des variables CSS.

De base, un site réalisé avec Gutenberg/FSE est déjà performant même sans optimisation (cache, etc...)

Exemple de code HTML sur un simple titre avec Elementor :

```
<div
  class="elementor-element elementor-element-3629291 elementor-widget elementor-widget-heading"
  data-id="3629291"
  data-element_type="widget"
  data-widget_type="heading.default"
>
  <div class="elementor-widget-container">
    <h2 class="elementor-heading-title elementor-size-default">
      Mon titre
    </h2>
  </div>
</div>
```


Code HTML pour le même titre avec Gutenberg :

```
<h2 class="wp-block-heading">Mon titre</h2>
```


4 - Réglages de l’interface pour guider l’utilisateur.rice

On peut régler assez finement certains réglages de l’interface :

- Les couleurs disponibles
- Les polices disponibles
- Les tailles des containers
- ....

Cela permet de garder une bonne cohérence entre les pages. L’éditeur de contenu est guidé et ne se retrouve pas perdu.

5 - Des éléments prêts à l'emploi dans l'interface

Avec les blocs et les patterns, l'éditeur de contenu dispose d'éléments disponibles et mis en forme.

Il suffit donc d'ajouter les éléments dans le contenu et juste d'éditer le contenu de cet élément. Simple et rapide.

Un gain de temps considérable et les utilisateurs se retrouvent plus confiant.

6 - Le thème peut être contenu dans des fichiers et pas uniquement en base

En tant que développeur, je peux créer un thème FSE basé sur des fichiers et déployer mon thème en production sans la base-de-donnée.

Vous pouvez conserver les fichiers sources sur un gestionnaire de versions et travailler d'une façon plus professionnelle.

7 - Réutilisation des éléments d'un site à l'autre

On peut se créer plusieurs blocs, patterns et templates afin d'optimiser la production de site.

On retrouve souvent les mêmes éléments d'un site à l'autre.

Exemple : FAQ. Je peux réutiliser tout mon système de FAQ sur différents sites.

8 - Parce que ...

- C’est présent dans WordPress et ça ne risque pas de changer
- Permet de livrer des sites professionnels
- Permet une grande autonomie des utilisateurs.rices/editeurs.rices
- WooCommerce passe de plus en plus sur les blocs.

Lexique du FSE

THEME.JSON

Le fichier de config pour les réglages du Gutenberg et du FSE + les styles par défaut des blocs.
Sans le savoir nous utilisons déjà le fichier theme.json de WordPress.

STYLES

Dérivés du theme.json pour proposer des styles différents pour un même theme.

LES TEMPLATES (MODÈLE DE CONTENU)

Comme le nom l'indique, il s'agit de template de page. Dans la création d'une page, nous avons soit le template par défaut, soit un template que l'on choisit dans le sélecteur.
Les templates sont 100% HTML.

LES TEMPLATES PARTS (ÉLÉMENTS DE MODÈLES)

Les parts représentent des éléments de template. Comme le header, le footer ou même des petits éléments comme le post-meta.html que l'on retrouve dans le thème twenty-twenty-three.
Les parts sont 100% HTML

LES PATTERNS (COMPOSITIONS)

Ce sont des sortes de composant HTML réutilisables. Des blocs HTML avec du style déjà assemblés disponibles directement dans l'éditeur Gutenberg.
Les patterns sont sous forme de fichier PHP mais nous verrons un peu plus tard que le rendu reste en mode HTML.

LES SYNCED PATTERNS (COMPOSITIONS SYNCHRONISÉES)

Ce sont des patterns réutilisables dans plusieurs contenus/pages et l'on peut modifier le pattern une seule fois pour toutes les pages. Uniquement disponible en passant par l'administration.

BLOCK

Ce sont les éléments principaux de l'éditeur Gutenberg. Dynamique ou statique, ils sont disponibles dans le sélecteur de bloc Gutenberg.

Ils disposent de paramètres (ou pas) sur la colonne de droite dans l'éditeur. Connaissez-vous la différence entre un bloc statique et un bloc dynamique ?

Un nouveau modèle mental

Les templates sont en HTML, logiquement, ils ne sont pas dynamiques. Cela oblige donc à réfléchir sous forme de bloc/pattern.

Dans une page, dans un template, dans un part, tout est bloc. Tout doit être pensé sous forme de bloc.

Pour les personnes qui font des Web Apps avec React/Vue, etc... c'est déjà une habitude de penser comme cela. Pour les autres, il faut se forcer à penser sous forme de bloc.

Il est possible de couvrir 80% des besoins avec les blocs de base présent dans WordPress

- block group, row, column
- block heading, paragraph, list, image, etc...
- block navigation, site logo, etc...
- query loop, post title, post content, etc...

PLUS AVANCÉ

- Block variations pour étendre les blocs de base (query loop, etc...)
- Plugin pour l'interface Gutenberg (sidebar, post status, etc...)

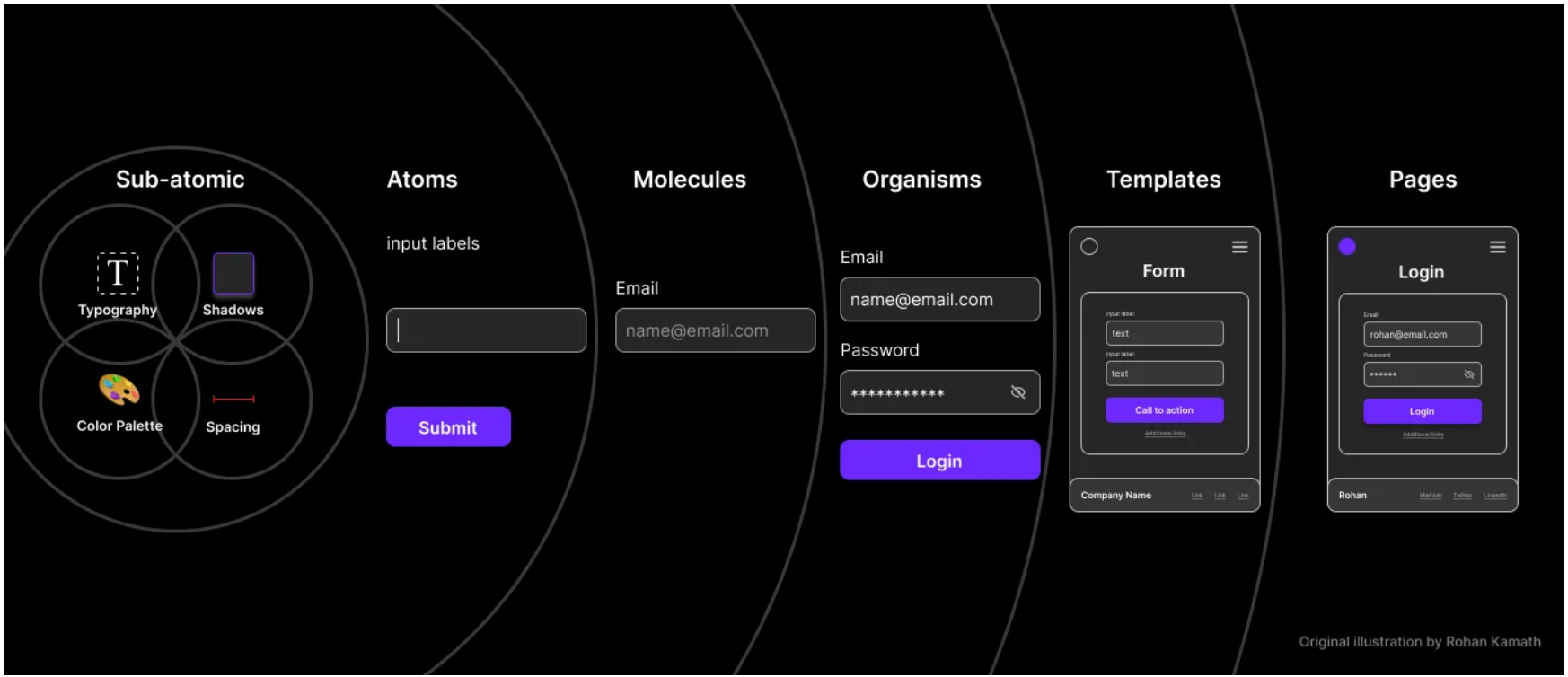
Avec le FSE, on ne pense plus sous forme de page mais sous forme de blocs

Stop les maquettes de pages

Atomic design

On peut aussi prendre l'atomic design comme modèle qui représente parfaitement le mode de pensée du FSE.

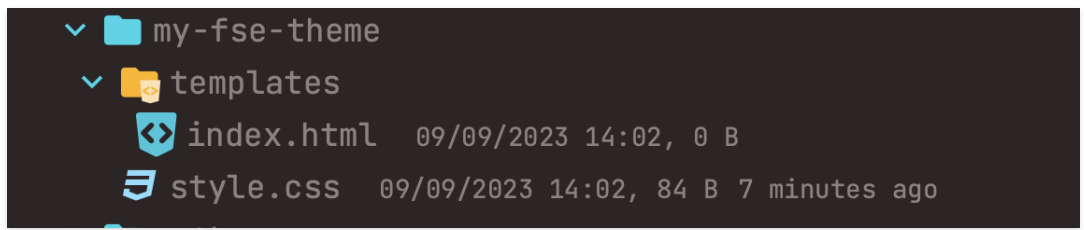
L'atomic design comprend des éléments sous forme d'atomes, de molécules, d'organismes, de template et de page. Vous voyez où je veux en venir ?


Sub-atomic
Typography
Shadows
Color Palette
Spacing
Atoms
input labels
Submit
Molecules
Email
name@email.com
Organisms
Email
name@email.com
Password
****
Login
Templates
Form
text
text
Call to action
Company Name
Pages
Login
rohan@email.com
Login
Rohan
Original illustration by Rohan Kamath

Getting started

Créer un Block Theme

Il suffit d'ajouter un fichier index.html dans un repertoire templates.



Demo

Ok, et après ?

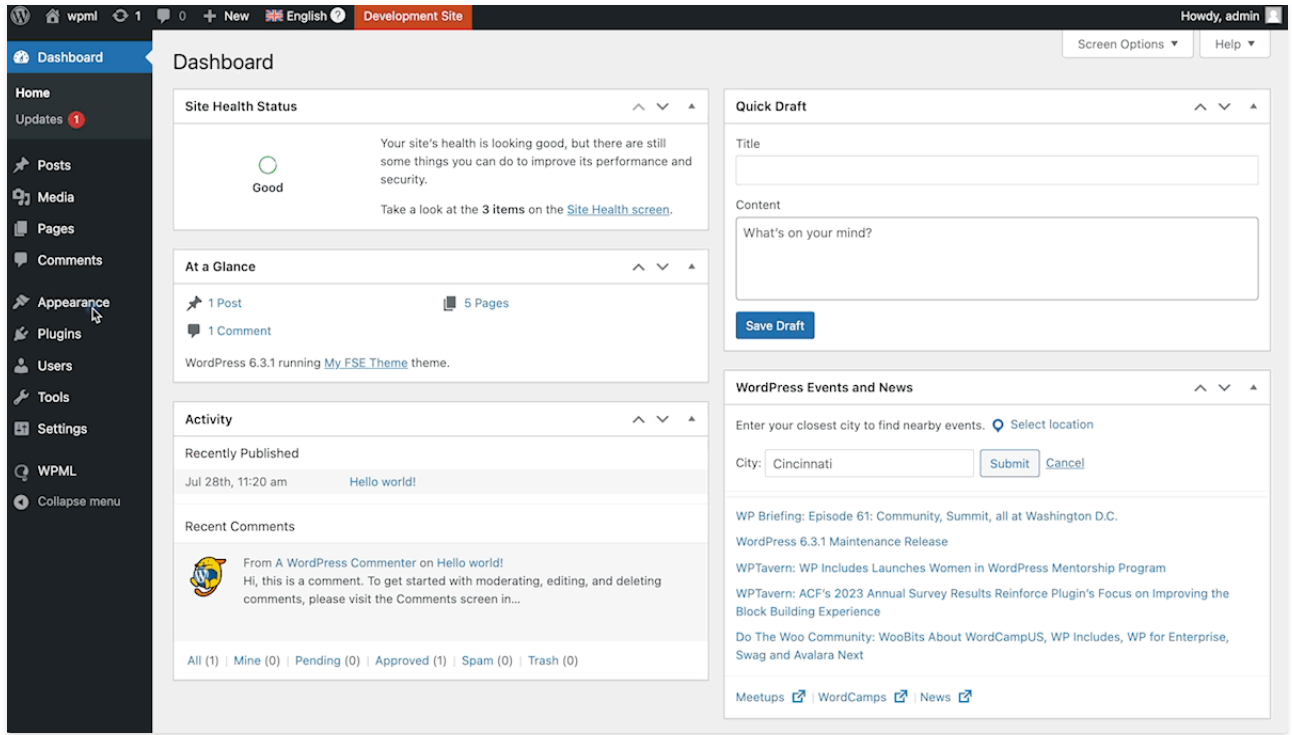
Méthode 1

Je créer mes élements directement dans l'interface de l'éditeur de site

- Templates
- parts (header, footer, etc...)
- patterns (compositions)

JE PEUX EXPORTER LE THÈME SOUS FORME DE FICHIER

Dans tous les panneaux d'édition de l'éditeur de site > Options > Export (Tools)



Méthode 2

Je fais directement dans mon éditeur de code tous les fichiers dans le thème (Mode je sais ce que je fais !)

From scratch ou en utilisant un Block Theme :

- https://olliewp.com/
- https://github.com/WordPress/twentytwentyfour

PS : J'ai le droit d'utiliser l'interface ;)

Code HTML des utilisés dans les templates, parts et patterns

Le code HTML représente des éléments (blocs, patterns, etc...). Les valeurs contenues dans les commentaires sont les réglages des éléments. Le code est parsé pour être rendu dans l'éditeur. En front les commentaires sont supprimés.

UN CONTAINER GROUP

```
<!-- wp:group {"layout":{"type":"constrained"}} -->
    <div class="wp-block-group">
    </div>
<!-- /wp:group -->
```

UN PARAGRAPHE

```
<!-- wp:paragraph -->
<p>Deserunt excepteur sunt consequat ad ea nulla ex.
Ut ullamco nostrud do exercitation id pariatur exercitation sunt qui est aliquip.</p>
<!-- /wp:paragraph -->
```

UN TITRE

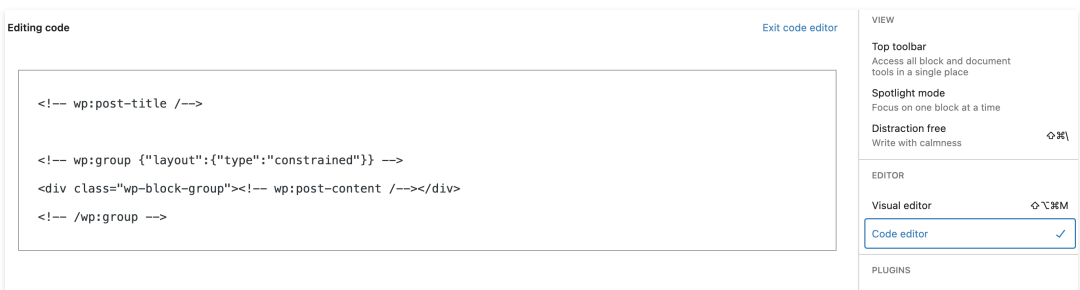
```
<!-- wp:heading {"textAlign":"center","fontSize":"3xlarge"} -->
<h2 class="wp-block-heading has-text-align-center has-3-xlarge-font-size">Ut ullamco nostrud do</h2>
<!-- /wp:heading -->
```


Comme je ne connais pas tous les élements, je peux utiliser l’interface de l’éditeur de site pour m’aider.

Je peux faire mes templates, templates part via l’interface et copier le code source.

Dans tous les panneaux d’édition de l’éditeur de site > Options > Copy all blocks (Tools)

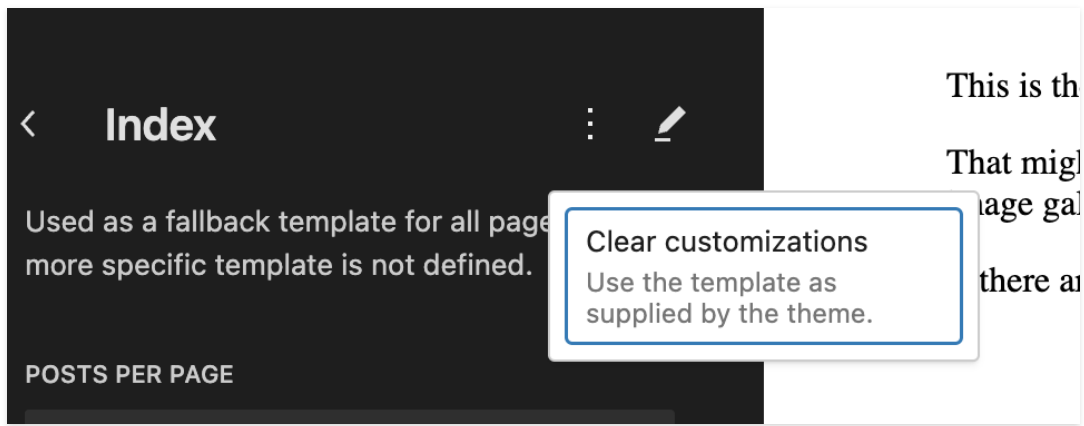
ou passer en mode "Code Editor" et copier la source.



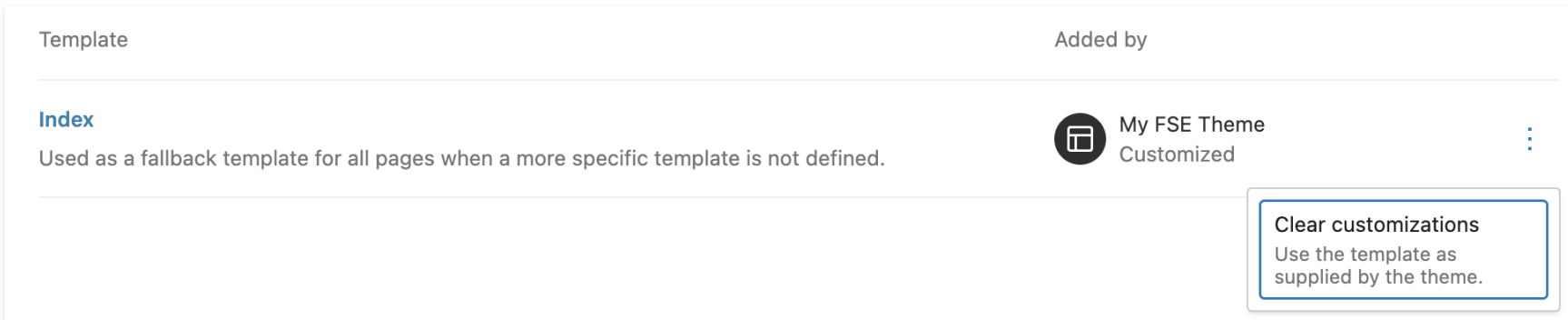
Comme je privilégie les versions fichiers (locales), je ne conserve pas les versions modifiées en base de données.

J'efface les customisations via :

- dans le panneau gauche de l'edition d'un élément



OU VIA L'INTERFACE DE GESTION DES TEMPLATES (OU PARTS, PATTERNS)



À retenir

- Toutes les modifications sont enregistrées en base de données
- Les fichiers du thème ne sont jamais modifiés
- Aucun fichier n’est créé
- On peut exporter le thème complet modifié avec la fonction disponible dans l’interface
- Pour prendre en compte le fichier et non la version DB, il faudra effacer les customisations

Toutes les customisations sont enregistrées dans la table `{prefix}posts`. Avec des post_types "wp_template_part, wp_template, wp_navigation, etc..."

Le fichier theme.json

Un des fichiers le plus important du FSE

PAR DÉFAUT, VOTRE WORDPRESS UTILISE DÉJÀ LE THEME.JSON QUI SE TROUVE DANS WP-INCLUDE.

si vous utilisez Gutenberg !

Il permet de configurer l'interface de Gutenberg et de mettre des styles par défaut.

```
{
    "$schema": "https://schemas.wp.org/wp/6.3/theme.json",
    "version": 2,
    "settings": {},
    "styles": {}
}
```

Spécifier la version est obligatoire.

Le `$schema` permet de valider le fichier et d'avoir l'autocomplétion.

La partie "settings"

Pour définir les réglages disponibles pour les blocs au niveau global

```
{
    "settings": {
        // réglages globaux
    },
}
```

ou au niveau des blocs pour les réglages spécifiques

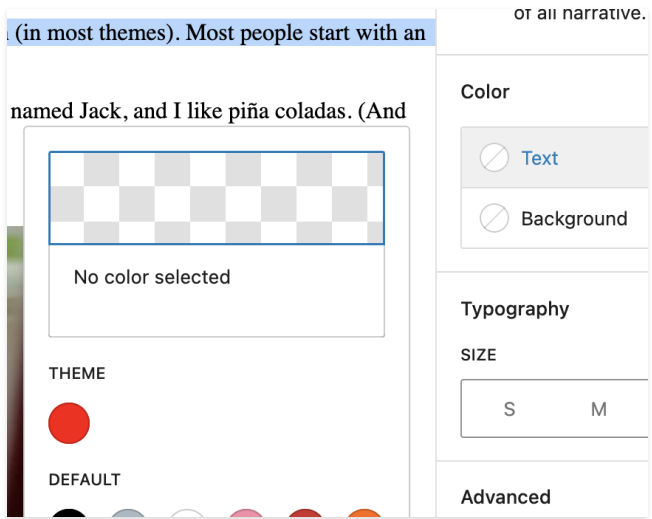
```
{
    "settings": {
        // réglages globaux
        "blocks": {
            // réglages par bloc
        }
    },
}
```


```
{
    "settings": {
        "color": {
            "palette": [
                {
                    "name": "Black",
                    "slug": "black",
                    "color": "#000000"
                },
            ]
        },
        "blocks": {
            "core/paragraph": {
                "color": {
                    "palette": [
                        {
                            "name": "Red",
                            "slug": "red",
                            "color": "#ff0000"
                        }
                    ]
                }
            }
        }
    }
}
```


Le systeme va donc générer des variables CSS propres à chaque bloc.

```css
body {
    --wp--preset--color--black: #000000;
}

.wp-block-paragraph {
    --wp--preset--color--red: #ff0000;
}
```



La partie "styles"

POUR DÉFINIR LES STYLES PAR DÉFAUT.

```
{
    "styles": {},
}
```

Comme pour la partie settings, nous avons des styles globaux et ensuite nous pouvons affiner par élements et blocs.

CE SONT DES STYLES PAR DEFAUT. VOUS POUVEZ LES MODIFIER DANS L'INTERFACE GUTENBERG AU MOMENT DE L'ÉDITION.

Premier niveau de styles

```
{
    "styles": {
        "border": {
            ...
        },
        "filter": {
            "duotone": "value"
        },
        "color": {
            "background": "value",
            "gradient": "value",
            "text": "value"
        },
        "spacing": {
            "blockGap": "value",
            "margin": {},
            "padding": {}
        },
        "typography": {}
    },
    ...
}
```


Le style pour les éléments

```
{
    "styles": {
        ...
        "link": {
                "border": {},
                "color": {},
                "spacing": {},
                "typography": {}
            },
            "h1": {},
            "h2": {},
            "h3": {},
            "h4": {},
            "h5": {},
            "h6": {}
        }
    },
    ...
}
```


Le style pour les blocs

```
{
    "styles": {
        ...
       "blocks": {
            "core/group": {
                "border": {},
                "color": {},
                "spacing": {},
                "typography": {},
                "elements": {
                    "link": {},
                    "h1": {},
                    "h2": {},
                    ...
                }
            },
            "etc": {}
        }
}
```


Pour les styles, la bonne pratique est de reprendre les variables générées par la partie settings.

```
{
    "styles": {
        "color": {
            "background": "var(--wp--preset--color--black)",
            "text": "var(--wp--preset--color--white)"
        }
    }
}
```


Quelques exemples de réglages

Layout

Quand le parent du bloc est en mode `"layout":{"type":"constrained"}`.

```
"layout": {
   "contentSize": "650px",
   "wideSize": "1200px"
  },
```

- contentSize : taille du contenu par défaut
- wideSize : taille du contenu en mode "wide"
- Reste le mode "full" qui prend toute la largeur de l'écran.

Typography (fontFamilies)

```
"typography": {
   "dropCap": false,
   "fluid": true,
   "fontFamilies": [
    {
     "fontFace": [
      {
       "fontFamily": "DM Sans",
       "fontStretch": "normal",
       "fontStyle": "normal",
       "fontWeight": "400",
       "src": [
        "file:./assets/fonts/dm-sans/DMSans-Regular.woff2"
       ]
      },
      ...
     ],
     "fontFamily": "\"DM Sans\", sans-serif",
     "name": "DM Sans",
     "slug": "dm-sans"
    },
   ]
  ...
}
```


La partie "fontFamilies" permet d’utiliser la Font API.

- Première chose, on retrouve les polices dans l’éditeur pour les éléments. Plutôt pratique
- La Font API charge la police uniquement si elle est utilisée ! Top pour la performance.

Attention fonctionne uniquement avec des polices locales.

 ~~6.4 ARRIVÉE DE LA FONT LIBRARY~~

Typography (fontSizes)

Tips => fluid: true

```
"typography": {
   "dropCap": false,
   "fluid": true,
   "fontFamilies": [
    {...},
   "fontSizes": [
    {
     "fluid": {
      "min": "0.875rem",
      "max": "1rem"
     },
     "size": "1rem",
     "slug": "small"
    },
    ...
   ]
```


La partie "fontSizes"

- Comme pour le reste, on retrouve les tailles de polices dans l'éditeur pour les éléments.
- On peut utiliser des tailles fluides (min et max). Pour s'adapter à la taille de l'écran.

Génère un code dans ce style :

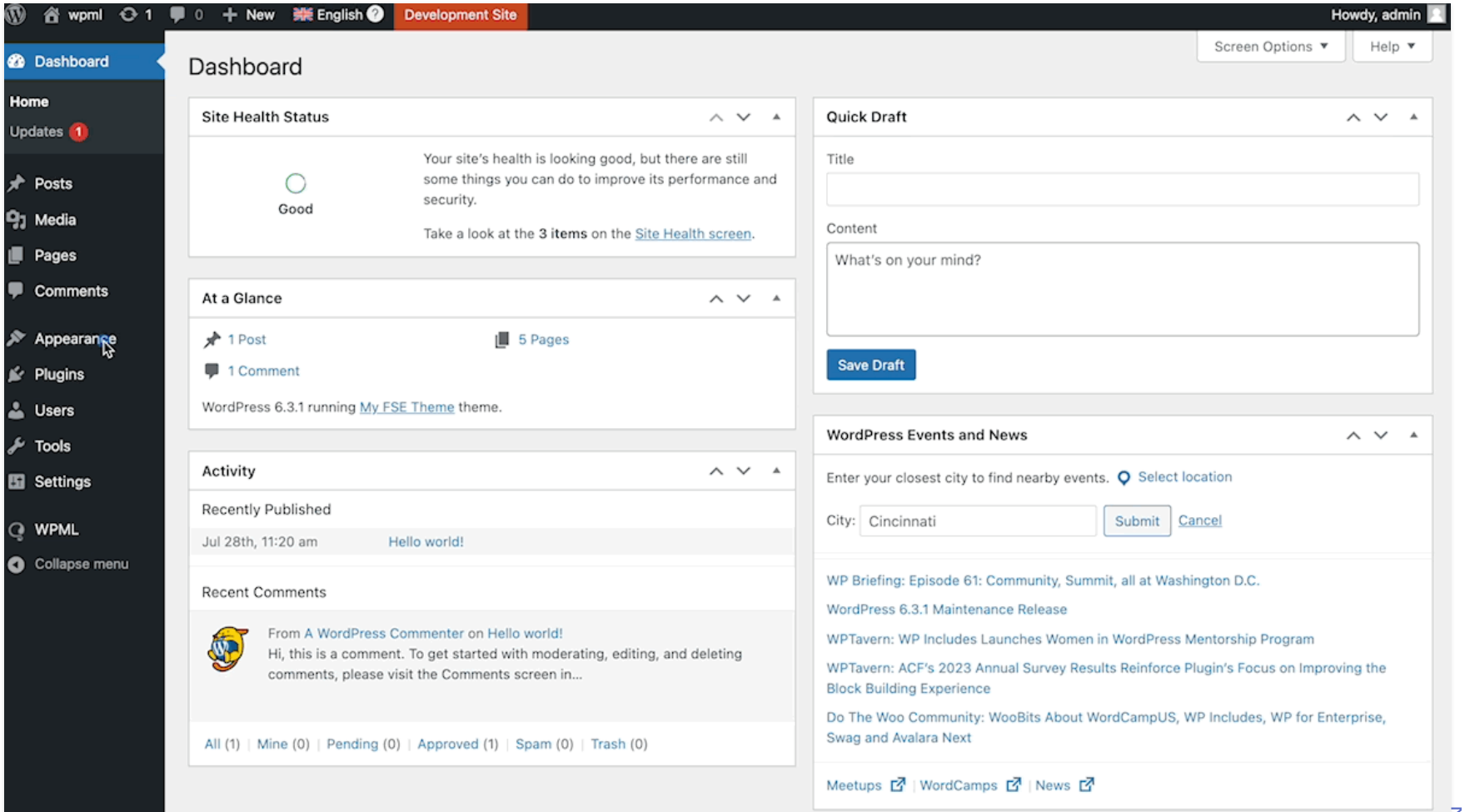
```
--wp--preset--font-size--medium: clamp(0.9rem, 0.9rem + ((1vw - 0.48rem) * 0.577), 1.2rem);
```


Trop compliqué le json !

Méthodes alternatives

Pour les réglages, il faut passer obligatoirement par le fichier json. Pas le choix ! Il faut donc se former un minimum au format json et aux valeurs possibles.

Par contre, pour le styles, vous pouvez utiliser l’interface "Styles" de l’éditeur de site. Faire vos styles pour les éléments et les blocs et ensuite exporter le fichier json.


wpml
1
0
New
English
Development Site
Howdy, admin
Dashboard
Home
Updates 1
Posts
Media
Pages
Comments
Appearance
Plugins
Users
Tools
Settings
WPML
Collapse menu
Dashboard
Screen Options
Help
Site Health Status
Good
Your site's health is looking good, but there are still some things you can do to improve its performance and security.
Take a look at the 3 items on the Site Health screen.
At a Glance
1 Post
5 Pages
1 Comment
WordPress 6.3.1 running My FSE Theme theme.
Activity
Recently Published
Jul 28th, 11:20 am
Hello world!
Recent Comments
From A WordPress Commenter on Hello world!
Hi, this is a comment. To get started with moderating, editing, and deleting comments, please visit the Comments screen in...
All (1) | Mine (0) | Pending (0) | Approved (1) | Spam (0) | Trash (0)
Quick Draft
Title
Content
What's on your mind?
Save Draft
WordPress Events and News
Enter your closest city to find nearby events.
Select location
City:
Cincinnati
Submit
Cancel
WP Briefing: Episode 61: Community, Summit, all at Washington D.C.
WordPress 6.3.1 Maintenance Release
WPTavern: WP Includes Launches Women in WordPress Mentorship Program
WPTavern: ACF's 2023 Annual Survey Results Reinforce Plugin's Focus on Improving the Block Building Experience
Do The Woo Community: WooBits About WordCampUS, WP Includes, WP for Enterprise, Swag and Avalara Next
Meetups
WordCamps
News


Guide de style
Enregistré
Styles
Texte
Média
trio Theme
Design
Widgets
Thème
Contenus embarqués
TITRES
Le code est une poésie
Le code est une poésie
Le code est une poésie
Aa
Typographie
Couleurs
Mise en page
Personnalisez l'apparence de blocs spécifiques pour l'ensemble du site.
Blocs
Page

Ou vous pouvez utiliser des generateurs de theme.json

Exemple :

- https://wpturbo.dev/generators/theme-json/
- https://www.gutenberg-devtools.com/generator/themejson-support

Dernière chose sur le theme.json

Passez en mode debug pour éviter le cache en mode developpement.

```
define('WP_DEBUG', true);
```

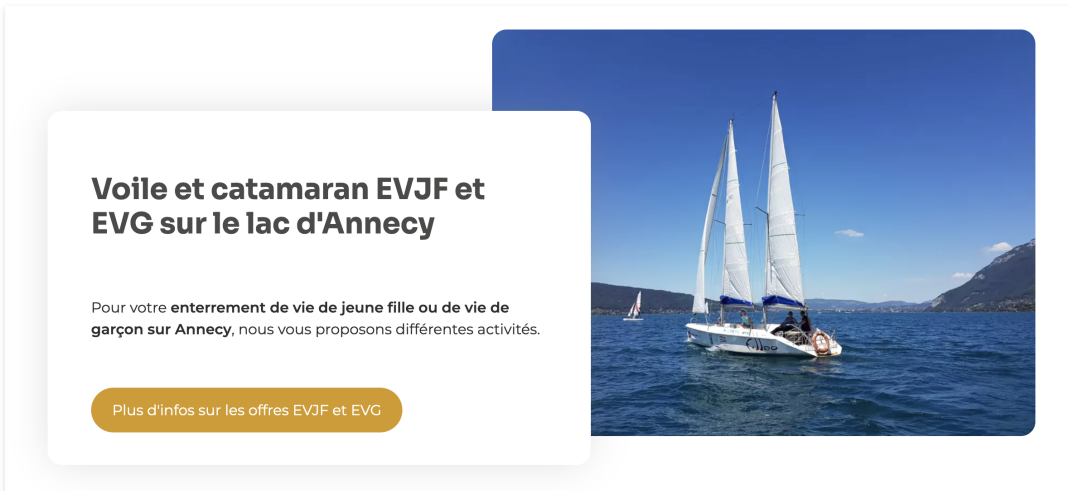

Les compositions (patterns)

Pour moi, c’est un élément qui change tout !

Sans les patterns, Gutenberg et le FSE seraient moins fun !

Les patterns sont des assemblages de blocs déjà stylisés. Ils sont disponibles dans le sélecteur de blocs > Compositions.

Permet à un utilisateur de créer un contenu rapidement et de façon harmonieuse.



Simple à créer et facile à utiliser

- Créer un pattern est plutôt accessible.
- Directement disponible dans le sélecteur (inserter).
- Preview du pattern dans le sélecteur.
- Inséré en un clic.
- Possibilité de le synchroniser pour modifier le pattern en un seul endroit.

Fichiers PHP avec des commentaires (obligatoire)

```php
<?php
/**
 * Title: Hidden 404
 * Slug: trio/hidden-404
 * Inserter: no
 */
?>
```

Inserter: no permet de ne pas afficher le pattern dans le sélecteur de blocs.

https://developer.wordpress.org/block-editor/reference-guides/block-api/block-patterns/#register_block_pattern

Des fichier PHP mais attention !

Oui, il est possible de mettre du code php dans les patterns mais dès qu'il est incorporé dans un contenu, il est transformé en HTML. Plus de partie dynamique !

Pareil, si il est dans un template ou un part. Si vous éditez le contenu de l'élément, le php est automatiquement transformé en statique (HTML).

Exemple : j'ai un pattern dans le fichier du footer dans mon thème. Si j'édite le footer, le pattern est transformé en HTML. Donc si j'ai du texte avec `__('mon texte', 'domaine');`. Il devient mon `mon text` et ne sera pas traduit.

Autoload

Les fichiers déposés dans `/patterns` sont disponibles automatiquement dans le sélecteur de composition.

Catégories

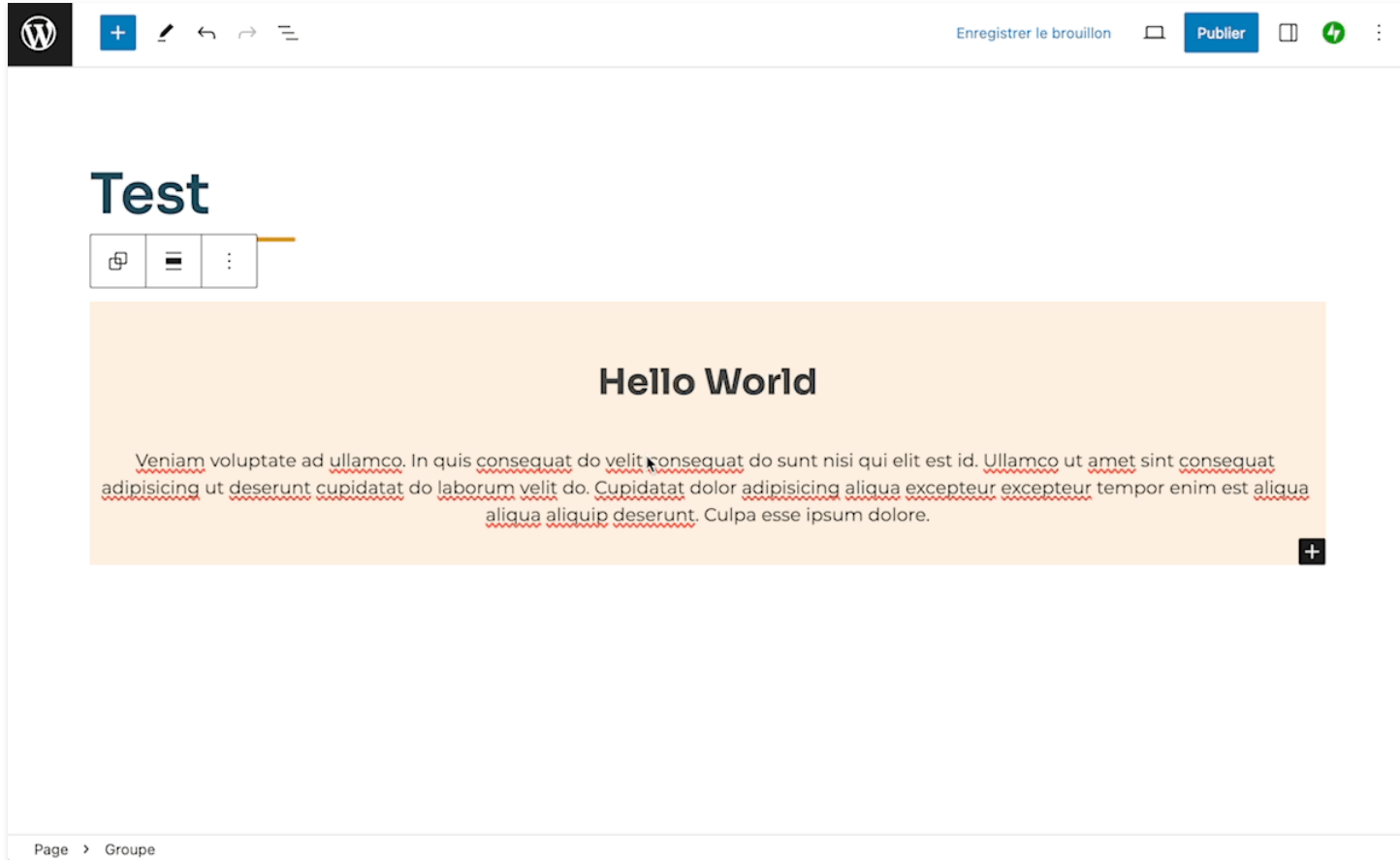
On peut utiliser les catégories existantes : Buttons, Columns, Featured, Gallery, Text, About, Call to action, Contact, Footers, Headers...

ou en créer pour notre thème.

```
if ( function_exists( 'register_block_pattern_category' ) ) {
    register_block_pattern_category(
      'mycategory',
      array(
            'label' => __( 'My Category', 'text-domain' ),
            'description' => __( 'Theme and more', 'text-domain' ),
       )
   );
}
```


Les patterns synchronisés

Si un pattern est utilisé dans plusieurs contenus, il est possible de le synchroniser. Pour le modifier à un seul endroit et que les modifications soient répercutées partout.


Enregistrer le brouillon
Publier
Test
Hello World
Veniam voluptate ad ullamco. In quis consequat do velit consequat do sunt nisi qui elit est id. Ullamco ut amet sint consequat adipisicing ut deserunt cupidatat do laborum velit do. Cupidatat dolor adipisicing aliqua excepteur excepteur tempor enim est aliqua aliqua aliquip deserunt. Culpa esse ipsum dolore.
Page › Groupe

Pattern Directory

Il est possible de le désactiver.

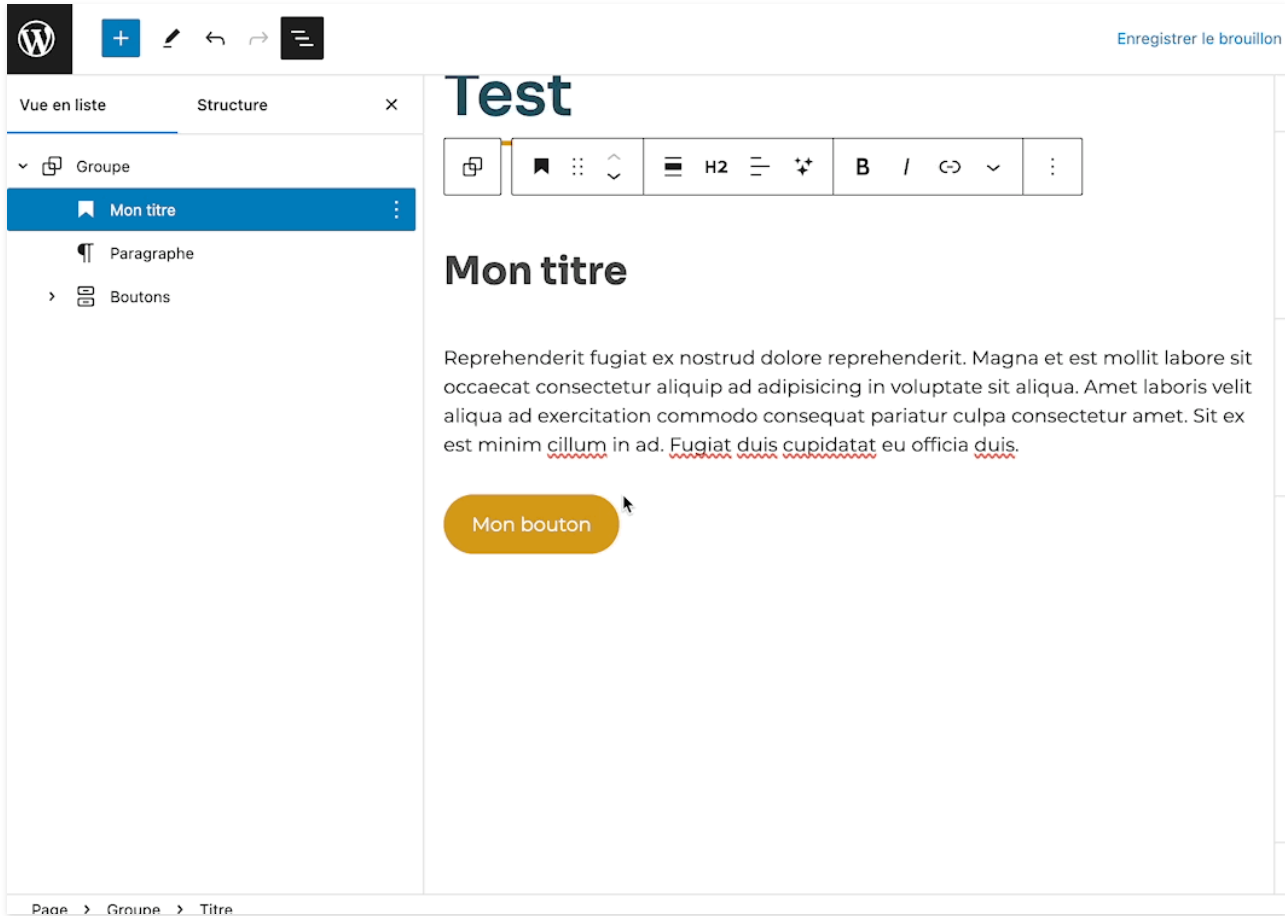
```
add_filter( 'should_load_remote_block_patterns', '__return_false' );
```


Lock

On peut verrouiller un pattern pour qu'il ne soit pas modifiable par l'utilisateur. L'utilisateur peut uniquement éditer le contenu.

Évite les mauvaises manipulations.

Verrouillage possible via l'interface ou dans le fichier.


Enregistrer le brouillon
Vue en liste
Structure
Groupe
Mon titre
Paragraphe
Boutons
Test
H2
Mon titre
Reprehenderit fugiat ex nostrud dolore reprehenderit. Magna et est mollit labore sit occaecat consectetur aliquip ad adipisicing in voluptate sit aliqua. Amet laboris velit aliqua ad exercitation commodo consequat pariatur culpa consectetur amet. Sit ex est minim cillum in ad. Fugiat duis cupidatat eu officia duis.
Mon bouton
Page
Groupe
Titre

Empêcher le déverrouillage par les éditeurs

```php
add_filter(
 'block_editor_settings_all',
 static function( $settings, $context ) {
  // Allow for the Editor role and above.
  $settings['canLockBlocks'] = current_user_can( 'delete_others_posts' );

  // Only enable for specific user(s).
  $user = wp_get_current_user();
  if ( in_array( $user->user_email, array( 'user@example.com' ), true ) ) {
   $settings['canLockBlocks'] = false;
  }

  // Disable for posts/pages.
  if ( $context->post && $context->post->post_type === 'page' ) {
   $settings['canLockBlocks'] = false;
  }

  return $settings;
 },
 10,
 2
);
```


Limites des patterns

Une fois insérer dans le contenu, le pattern devient un simple bloc. Et donc on ne peut plus le modifier globalement.

Il n'y a pas de solution simple pour modifier en masse des patterns déjà inseré dans le contenu.

Alternative :

- Utiliser des blocs réutilisables
- Utiliser des Templates parts
- Utiliser des blocs

Les templates

Comme déjà mentionné, ils sont **100% HTML**.

La hierarchie des templates reste la même

J’espère que vous connaissez la hiérarchie des templates de WordPress !


Page d'archive
Archive d'auteur·ice
Archive de catégorie
Archive de type de publication
Archive de taxonomie
Archive de date
Archive d'étiquette
author-$nicename.php
category-$slug.php
taxonomy-$taxonomy-$term.php
Archive d'une année
Archive d'un mois
Archive d'un jour
tag-$slug.php
author-$id.php
category-$id.php
taxonomy-$taxonomy.php
date.php
tag-$id.php
author.php
category.php
archive-$posttype.php
taxonomy.php
date.php
tag.php
archive.php
index.php
Publication seule
Publication seule (articles natifs, types de publications personnalisées, fichiers attachés)
Page statique
Fichier attaché
Type de publication personnalisé
Articles natifs
Modèle de page
$mimetype-$subtype.php
Modèle personnalisé
Modèle par défaut
$subtype.php
If selected: $custom.php
If selected: $custom.php
$custom.php
$mimetype.php
single-$posttype-$slug.php
page-$slug.php
attachment.php
single-$posttype.php
single-post.php
page-$id.php
single.php
page.php
singular.php
Page d'accueil
front-page.php
Affichage d'une page
Affichage du blog
Accueil du blog
home.php
Erreur 404
404.php
Recherche
search.php
Contenus embarqués :
embed-{post-type}-{post_format}.php →
embed-{post-type}.php →
embed.php →
wp-includes/theme-compat/embed.php
Modèle principal
Modèle secondaire
Modèle dynamique
Type d'écran

Il faut garder des templates assez minimalistes

Reporter les gros morceaux de code dans des patterns ou des templates parts.

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

<!-- wp:group {"tagName":"main"} -->
<main class="wp-block-group">
    <!-- wp:group {"layout":{"type":"constrained"}} -->
    <div class="wp-block-group">

        <!-- wp:pattern {"slug":"prefix/hero"} /-->

        <!-- wp:post-title {"level":1} /-->
    </div>
    <!-- /wp:group -->

    <!-- wp:post-content {"layout":{"type":"constrained"}} /-->

    <!-- wp:template-part {"slug":"comments","tagName":"section"} /-->

</main>
<!-- /wp:group -->

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


Exemple de code pour intégrer des patterns, template parts ou block

Pattern:

```
<!-- wp:pattern {"slug":"prefix/hero"} /-->
```

Template part:

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

Block:

```
<!-- wp:gravityforms/form {"formId":"2","title":false,"description":false,"ajax":true,"inputPrimaryColor":"#204ce5"} /-->
```


Les blocs

Bloc statique ou dynamique ?

- Un bloc statique est un bloc qui une fois sauvé, le contenu est figé (statique) :
  - Un titre
  - Un paragraphe
  - Une image
  - Un bouton
  - ...
- Un bloc dynamique possède un rendu serveur. Une fonction PHP génère le contenu a afficher :
  - Liste de posts
  - Elements dynamiques

Quand créer un bloc ?

- Quand on doit afficher un champ personnalisé (ACF)
- Si on veut créer un élément reutilisable sur plusieurs projets (FAQ, Carousel, ...)
- Quand on veut afficher un code HTML différent de la norme Gutenberg/FSE

Comment créer un bloc ?

- Apprendre JavaScript, React JS !
- Installer Node JS, etc...

```
npx @wordpress/create-block@latest todo-list
```

Clairement, pas à la porté de tout le monde !

Advanced Custom Fields

ACF permet de créer des blocs assez facilement :

- Les blocs sont uniquement dynamiques.
- Les champs de réglages s'affichent dans la colonne de droite de l'éditeur.
- L'expérience est pas aussi bonne que les blocs natifs mais ça fait le job.

Par contre, cela créer une dépendance avec ACF.

i18n et le FSE

Pas encore natif.

Prévu dans la phase 4 de la feuille de route à long terme.

Polylang

PREND EN CHARGE LE FSE EN MODE PRO UNIQUEMENT

J’ai donc créé une solution pour gérer le FSE en mode free 😉

J’ajoute des templates avec un suffixe de langue

Exemple : page-fr.html

Et j’ajoute des templates dans le tableau des templates recherchés par WordPress.

```
add_filter('frontpage_template_hierarchy', 'add_language_templates', 10, 1);
add_filter('home_template_hierarchy', 'add_language_templates', 10, 1);
add_filter('index_template_hierarchy', 'add_language_templates', 10, 1);
add_filter('page_template_hierarchy', 'add_language_templates', 10, 1);
...
```


Mon hack pour Polylang

```php
function add_language_templates($template_file)
{
  // new template files array
  $template_file_lg = [];
  // get current language
  $lang = apply_filters('gm_current_language', null);
  // template defined in editor
  $template = get_page_template_slug();
  if($template) {
    $template_file[] = $template;
  }
  foreach ($template_file as $key => $value) {
    // add before template file name with the lang pattern = index-{lang}.php
    if ($lang !== '') {
      $template_file_lg[] = str_replace('.php', '-' . $lang . '.php', $value);
    }
    $template_file_lg[] = $value;
  }
  if ($lang !== '') {
    $template_file_lg[] = 'index-' . $lang . '.php';
  }
  $template_file_lg[] = 'index.php';
  return $template_file_lg;
}
```

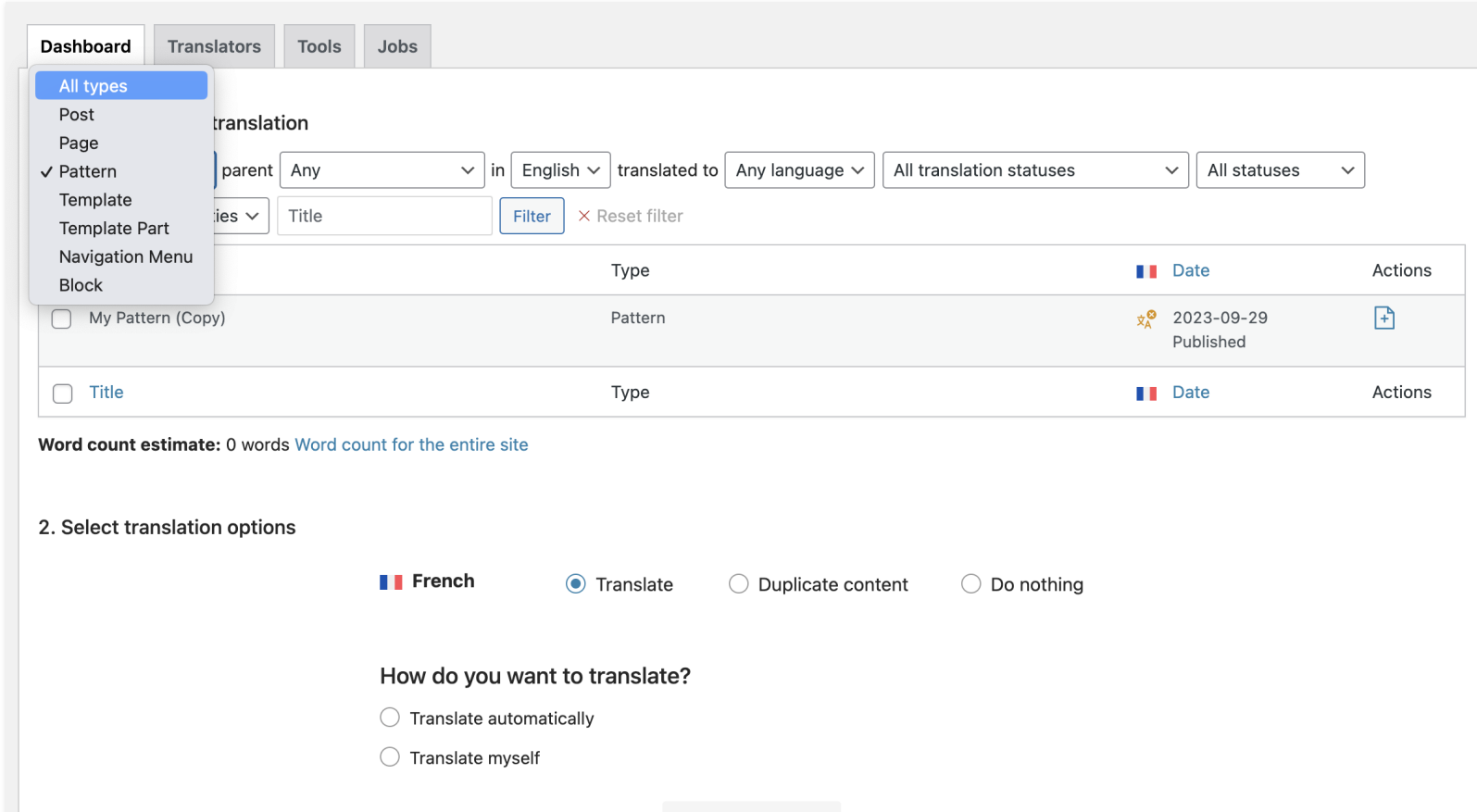

WPML

Prend en charge le FSE !

Gère la traduction des templates, pattern, block, navigation, etc...

Par contre, il ne trouve pas les éléments fichiers. Il faut modifier et sauver en base pour les voir disponibles dans l'interface de traduction.

Pas simple à gérer.


Dashboard
Translators
Tools
Jobs
All types
Post
Page
Pattern
Template
Template Part
Navigation Menu
Block
translation
parent
Any
in
English
translated to
Any language
All translation statuses
All statuses
Title
Filter
Reset filter
Type
Date
Actions
My Pattern (Copy)
Pattern
2023-09-29
Published
Title
Type
Date
Actions
Word count estimate: 0 words Word count for the entire site
2. Select translation options
French
Translate
Duplicate content
Do nothing
How do you want to translate?
Translate automatically
Translate myself

Les librairies et ressources

Ressources

Documentation officielle : https://developer.wordpress.org/block-editor/

Sites :

- https://fullsiteediting.com
- https://gutenberg.10up.com/

Block Library :

- https://wordpress.org/plugins/layout-grid/
- https://github.com/godaddy-wordpress/coblocks

Thèmes :

- https://olliewp.com/
- https://generateblocks.com/
- https://www.kadencewp.com/blog/introducing-blocks-3/
- https://wpspectra.com/

Figma :

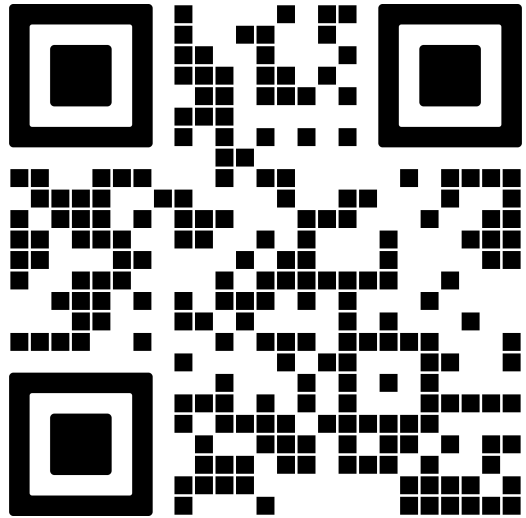
- https://johannes-wp.com/

À venir

Le prochain thème par défaut de WordPress (6.4) :

https://make.wordpress.org/core/2023/08/24/introducing-twenty-twenty-four/

Merci !



https://wp-biarritz-23.goodmotion.fr