Documentation thème FSE

Table des matières

Un thème FSE, c’est quoi ?

Le thème Full Site Editing, ou thème FSE, est un nouveau type de thème WordPress qui change pas mal la manière dont on construit un site. Au lieu de passer par plein de fichiers PHP et des méthodes parfois un peu rigides, il utilise l’éditeur de blocs pour permettre de personnaliser tout le site, directement depuis une interface visuelle.

Avec un thème FSE, on ne travaille plus uniquement sur le contenu des pages, mais aussi sur la structure globale : l’en-tête, le pied de page, les pages d’archives, et même la façon dont les articles s’affichent. Tout ça devient modifiable avec des blocs, un peu comme on construit un puzzle.

Pour ça, le thème est organisé différemment. Il repose beaucoup sur des fichiers modèles en HTML, des morceaux de modèles appelés "parts", et un fichier important appelé “theme.json” qui gère les styles et réglages globaux. Ce système rend la création et la personnalisation plus visuelle, plus intuitive, et surtout plus flexible.

En résumé, un thème FSE propose une approche moderne qui permet à la fois aux développeurs et aux utilisateurs d’avoir plus de contrôle sur leur site, tout en rendant la personnalisation accessible directement depuis WordPress.

Son fonctionnement

Le Full Site Editing fonctionne autour d'un principe simple : utiliser les blocs, déjà connus pour créer du contenu, pour construire toutes les parties du site. Ça veut dire que peu importe qu'il s'agisse de la page d'accueil, le menu, du pied de page, ou même de la présentation des articles, tout peut être monté, modifié et personnalisé avec des blocs.

Concrètement, un thème FSE met à disposition plusieurs fichiers modèles (templates) et morceaux de modèles (template parts) en HTML. Ces fichiers définissent la structure de chaque page ou section du site. Ensuite, l'éditeur de site WordPress permet d'ouvrir ces modèles en mode visuel et de les modifier directement, sans toucher au code.

Tout ce qui est réglages de styles, couleurs, typographies ou marges, est géré dans un fichier central appelé "theme.json". Ce fichier fait office de guide pour l'éditeur et assure que tout reste cohérent sur l'ensemble du site.

C'est un vrai changement par rapport aux thèmes classiques, car avant, beaucoup de ces modifications nécessitaient de coder dans des fichiers PHP, CSS ou JavaScript. Avec le FSE, on a une interface intuitive qui facilite la personnalisation, tout en gardant la puissance et la flexibilité des thèmes traditionnels.

Finalement, avec le FSE, créer un site WordPress devient une expérience bien plus graphique et intuitive, où tout peut être ajusté sans difficulté.

Qu’est ce qui le compose ?

Un thème Full Site Editing est constitué de plusieurs fichiers et dossiers qui travaillent ensemble pour construire le site. Chaque élément a son rôle : certains définissent la structure des pages, d’autres gèrent les styles, ou encore les ressources comme les images et les polices.

Contrairement aux thèmes classiques, on trouve dans un thème FSE beaucoup de fichiers HTML qui servent de modèles et de parties de modèles, ainsi que le fichier “theme.json”, qui centralise les réglages globaux. On retrouve aussi des fichiers traditionnels comme “functions.php” ou encore “style.css”, mais leur rôle évolue un peu.

Dans la suite, on va détailler chacun de ces éléments pour mieux comprendre leur utilité et leur fonctionnement.

index.php

Le fichier “index.php” est souvent considéré comme le fichier principal d’un thème WordPress, y compris dans un thème FSE. Il sert de base de secours si aucun autre modèle plus spécifique n’est trouvé pour afficher une page. Par exemple, si aucune page, article, archive ou autre template n’est défini, WordPress utilisera “index.php” pour afficher le contenu.

Dans un thème Full Site Editing, le rôle de “index.php” est parfois réduit, car la structure des pages est plutôt gérée par des fichiers HTML dans les dossiers templates et parts” Cependant, “index.php” reste obligatoire pour que le thème soit valide et puisse être activé. Souvent, il contient simplement une ligne minimale qui appelle la fonction WordPress pour charger l’environnement du thème.

En résumé, même si l’essentiel du rendu se fait ailleurs dans un thème FSE, “index.php” reste un point d’ancrage indispensable, garantissant que WordPress saura toujours comment s’appuyer sur le thème.

screenshot.png

Le fichier "screenshot.png" est l'image qui représente visuellement le thème dans l'interface d'administration de WordPress, notamment dans la liste des thèmes disponibles. C'est ce visuel qui permet d'identifier rapidement un thème parmi d'autres.

Pour qu'elle soit bien affichée partout, cette image doit respecter certaines dimensions : idéalement 1200 pixels de large sur 900 pixels de haut. Ce format offre un bon compromis entre qualité et rapidité de chargement dans le tableau de bord.

Il est important que cette image soit claire, attractive et qu'elle reflète l'identité ou le style du thème. Cela aide à donner une première impression positive à l'utilisateur qui cherche un thème à activer.

functions.php

Dans un thème FSE, le fichier "functions.php" garde un rôle important, même s'il est moins central que dans les anciens thèmes classiques. Il permet surtout d'ajouter des fonctionnalités spécifiques au thème, sans toucher au cœur de WordPress.

Ce fichier fonctionne comme une boîte à outils : on y écrit du code PHP pour activer certaines options, enregistrer des blocs, ajouter des filtres, ou encore inclure des fichiers supplémentaires.

Voici quelques usages typiques :

- Chargement des fichiers CSS/JS personnalisés (même si avec le FSE, la gestion des styles passe plutôt par "theme.json", on peut encore avoir besoin de charger des assets manuellement dans certains cas).
- Ajout de blocs personnalisés via des plugins ou du code maison.
- Déclaration de zones de menus ou de fonctionnalités spécifiques (même si les menus sont maintenant gérés par les blocs de navigation).
- Activation de certaines fonctionnalités de WordPress avec add_theme_support().

Voici un exemple simple d'un "functions.php" de thème FSE :

```
<?php

function mon_theme_scripts()

 { wp_enqueue_style( 'mon-theme-style', get_template_directory_uri() . '/style.css' ); }

add_action( 'wp_enqueue_scripts', 'mon_theme_scripts' );
```

style.css

Le fichier "style.css" dans un thème FSE n'a plus le même rôle central que dans les anciens thèmes WordPress, mais il reste important à plusieurs niveaux.

Son premier objectif est informatif : il contient l'en-tête du thème, c'est-à-dire les métadonnées que WordPress utilise pour reconnaître et afficher les informations du thème dans l'administration (nom, auteur, version, etc.). Ce bloc d'informations doit absolument se trouver au début du fichier pour que le thème soit reconnu correctement.

Au-delà de ça, "style.css" peut toujours être utilisé pour ajouter des règles CSS personnalisées, même si la gestion des styles globaux est désormais largement prise en charge par le fichier "theme.json". Il peut donc servir à ajouter des ajustements visuels, des classes spécifiques, ou du CSS que le système de blocs ne permet pas encore de gérer nativement.

Enfin, ce fichier est aussi celui que WordPress charge par défaut dans certaines circonstances, notamment pour l'aperçu du thème. Même s'il est vide ou presque, il est donc toujours conseillé de l'inclure.

Voici un exemple d'en-tête d'un fichier "style.css" :

```
/*
 Theme Name: Mon Thème FSE
Author: Nom de l'auteur
Version: 1.0
Requires at least: 6.0
*/
```

theme.json

"theme.json" est le fichier central d'un thème Full Site Editing. C'est ici que tout se joue : design, structure, réglages, comportements… Il permet de contrôler presque tous les aspects du site, sans écrire une ligne de CSS ou de PHP.

Grâce à ce fichier unique, on peut définir comment le site doit apparaître et se comporter dans l'éditeur de blocs, tout en garantissant une cohérence globale. Voici les principales sections que l'on y trouve :

version

Ce champ est obligatoire. Il indique la version du schéma JSON utilisé par WordPress pour interpréter les réglages du thème. La version est actuelle est la 3.

Il est important d'utiliser la bonne version, car elle détermine les fonctionnalités disponibles.

La version 3 permet notamment une meilleure gestion des styles par bloc, des contrôles plus fins sur l'interface de l'éditeur, ainsi qu'un support plus avancé pour les thèmes enfants.

settings

Cette section permet de configurer les capacités de personnalisation disponibles dans l'éditeur de site.

Par exemple, on peut y :

- Définir une palette de couleurs ou des tailles de police,
- Activer ou désactiver des options comme les espacements, les alignements, les bordures, etc,
- Et bloquer certaines fonctionnalités pour garder le contrôle sur la mise en page.

C'est ici qu'on déclare tout ce qui sera proposé dans l'interface à l'utilisateur.

styles

Dans cette partie, on définit les styles globaux du site. Ces styles seront appliqués par défaut, sauf s'ils sont surchargés par l'utilisateur ou un bloc.

On y déclare par exemple :

- La couleur de fond du site,
- La couleur du texte,
- La typographie par défaut,
- Et des styles spécifiques à des blocs précis (par exemple : styliser tous les titres "h2" de la même façon).

C'est un peu l'équivalent d'un gros fichier CSS, mais sous forme de JSON, interprété automatiquement par WordPress.

Ce fichier est donc essentiel à la création d'un thème FSE cohérent. C'est le point de départ de toute personnalisation sérieuse, et une vraie avancée vers un WordPress plus moderne, plus lisible, et plus modulaire.

Avant, il fallait jongler entre CSS, PHP, JavaScript et plein d'options dans les extensions. Maintenant, tout est centralisé dans ce fichier unique, lisible, maintenable, et pris en charge nativement par WordPress. Et surtout : ce qui est déclaré dans "theme.json" est directement visible et modifiable dans l'éditeur de site, sans code supplémentaire.

Bref, impossible d'imaginer un thème FSE sans lui. C'est le fichier à maîtriser en priorité quand on veut créer un thème moderne et cohérent

assets

Le dossier "assets" est utilisé pour organiser toutes les ressources front-end du thème. Ce n'est pas une obligation imposée par WordPress, mais c'est une convention très utile pour garder une structure propre et logique. C'est dans ce dossier qu'on place ce qui sera chargé côté utilisateur (polices, styles CSS supplémentaires, images, etc.).

Il est généralement divisé en plusieurs sous-dossiers selon le type de fichiers. Voici les plus courants :

fonts

Ce sous-dossier contient les fichiers de polices personnalisées utilisées dans le thème (ex: .woff, .woff2, .ttf, etc.).

Depuis WordPress 6.4, il est possible de déclarer des polices locales directement dans le theme.json, ce qui permet de les utiliser facilement dans les styles globaux ou spécifiques à certains blocs.

Il est conseillé de toujours utiliser le format .woff2, quand cela est possible, car il est plus léger et le plus optimisé pour le web. Par facilité, il est aussi possible de créer un sous-dossier pour chaque police ce qui permet de les trier plus facilement par ordre alphabétique.

css

Même si le theme.json permet de gérer énormément de styles (typographie, couleurs, espacements...), on peut toujours ajouter des fichiers CSS externes si besoin, pour des styles plus complexes ou spécifiques.

Par exemple :

- Des classes utilitaires personnalisées,
- Des animations CSS,
- ...

Ces fichiers peuvent être chargés manuellement depuis le fichier "functions.php" si nécessaire.

styles

Ce sous-dossier est utilisé dans certains thèmes FSE pour organiser des variantes de styles (aussi appelées style variations). Chaque fichier .json présent ici représente une variation de style du thème principal (couleurs, typographie, espacement…). Ces variantes sont affichées directement dans l'interface de personnalisation de WordPress, ce qui permet à l'utilisateur de changer rapidement de style global. Chaque variation suit la même structure que le theme.json, mais ne contient que les paramètres à surcharger.

Dans ce dossier, il est aussi possible de créer un sous-dossier nommé "blocks" qui lui permettra alors d'enregistrer des variantes pour des éléments spécifiques, comme par exemple si vous avez plusieurs sortes de boutons au sein d'un même site, c'est ici que vous pourrez définir vos variations.

Le dossier "assets" n'est donc pas obligatoire, mais il devient très vite indispensable pour garder un thème clair, bien structuré et facilement maintenable. Il permet de centraliser toutes les ressources statiques du thème, tout en restant compatible avec les bonnes pratiques du Full Site Editing.

templates

Le dossier "templates" contient les modèles de pages qui définissent la structure globale des différentes vues d'un site WordPress. Ces fichiers sont essentiels : ce sont eux qui permettent d'afficher correctement la page d'accueil, un article, une page statique, une archive, une page 404, etc. Chaque fichier est écrit en HTML et repose entièrement sur la logique des blocs Gutenberg, lisibles sous la forme de commentaires (<!-- wp:... -->).

Le fichier de base, "index.html", est obligatoire : il sert de modèle de secours si aucun autre modèle plus spécifique n'est trouvé. À côté de lui, on peut retrouver d'autres fichiers comme "page.html" (pour les pages), "single.html" (pour les articles), "archive.html" (pour les archives de contenu), ou encore "404.html" pour la page d'erreur. Ces derniers ne sont pas nécessaires à la création d'un thème. WordPress applique une logique de hiérarchie de modèles : si un modèle plus précis existe (par exemple "single-post.html" pour les articles), il sera utilisé en priorité. Sinon, le système remontera la hiérarchie jusqu'à tomber sur "index.html". Cela permet de personnaliser finement chaque type de contenu sans créer un doublon de structure.

Tous ces fichiers peuvent être créés ou modifiés dans l'éditeur de site, puis exportés au format HTML. Ils peuvent aussi être écrits manuellement si on préfère une approche plus "développeur". Dans tous les cas, ils représentent une évolution majeure par rapport aux anciens fichiers .php, en rendant la construction des pages visuelle, modulaire et entièrement pilotée par blocs.

parts

Le dossier "parts" contient ce qu'on appelle les parties de modèles (*template parts* en anglais). Ce sont des morceaux de page réutilisables qui permettent de construire des templates plus facilement et de manière plus modulaire. Par exemple, un en-tête, un pied de page ou une sidebar peuvent être créés une seule fois ici, puis appelés dans plusieurs modèles différents.

Comme pour les modèles, chaque fichier ici est un fichier .html contenant des blocs Gutenberg. On y retrouve souvent des noms comme "header.html", "footer.html" ou "sidebar.html", mais ce ne sont pas des noms obligatoires : on peut les nommer librement, du moment qu'on les appelle correctement depuis les templates.

L'intérêt est double :

- Centraliser certaines zones récurrentes du site (évite les duplications) ;
- Permettre leur modification rapide depuis l'éditeur de site, sans toucher aux fichiers un par un.

Dans un thème FSE, ces parties sont gérées exactement comme les modèles principaux : elles peuvent être modifiées visuellement dans l'interface, ou manuellement dans le code. On peut aussi les déclarer dans le "theme.json" pour en fixer certains comportements (comme leur position ou leur verrouillage).

L'utilisation des "parts" permet d'avoir un site bien structuré, plus facile à maintenir et à faire évoluer dans le temps.

En résumé

Un thème Full Site Editing s’appuie sur une organisation claire de fichiers et dossiers qui travaillent ensemble pour créer un site web complet. Le fichier central est "theme.json", qui gère les styles, les mises en page et les options du thème. Les autres fichiers comme "index.php" et "functions.php" sont surtout là pour assurer la compatibilité avec WordPress.

Les dossiers comme "templates" et "parts" permettent de structurer le site en modèles de pages et en parties réutilisables comme l’en-tête ou le pied de page. Enfin, "assets" regroupe toutes les ressources visuelles et styles complémentaires.

Cette structure rend les thèmes FSE à la fois simples à personnaliser et puissants, en mettant l’accent sur la modularité et la flexibilité, que ce soit pour les développeurs ou les utilisateurs.