CRÉER UN THÈME WORDPRESS

à partir d'un modèle statique HTML/CSS

Bienvenue !

Une feuille de route

Ce document est un résumé du process que j'utilise souvent pour créer un thème WordPress à partir d'un template statique HTML/CSS. Certaines étapes peuvent être interverties, et certaines sont plus longues que d'autres, mais l'essentiel y est pour comprendre comment bâtir un thème de A à Z.

Les fichiers d'accompagnement

Ce document accompagne l'atelier que j'ai eu l'honneur de présenter au WC Paris 2018.
Si vous ne les avez pas déjà, les fichiers du thème présenté peuvent être téléchargés sur Github:

https://github.com/vincedubroeucq/wcparis2018

Mettre en place le thème dans l'administration

Créez un dossier pour votre thème

Créez un dossier au nom de votre thème dans le dossier **wp-content/themes/** de votre installation de WordPress.

Créez les fichiers essentiels

Dans ce dossier, créez deux fichiers **index.php** et **style.css**.
Dans **style.css**, ajoutez le bloc de commentaires qui va permettre à WordPress de reconnaître le thème.

Mettre en place le thème dans l'administration

Activez votre thème

Si l'entête de **style.css** contient au moins **"Theme Name:"**, le thème devrait apparaître dans l'administration. Vous pouvez désormais l'activer.

Copiez-collez (salement) le contenu du prototype

Copiez-collez le contenu du fichier **index.html** dans **index.php**, et les styles de **style.css**, en conservant le bloc de commentaires en entête, bien entendu.

Organiser les fichiers de base du thème

Header, footer, et sidebar

Créez trois nouveaux fichiers: **header.php**, **footer.php** et **sidebar.php**.
Ces fichiers vont contenir le code nécessaire pour afficher l'entête du site, le pied de page et la barre latérale du blog, respectivement.

Coupez-collez (encore salement) le contenu de index.php

Coupez le contenu de **index.php**, depuis la déclaration **<!doctype>** jusque la balise ouvrante de la zone de contenu,et collez tout ça dans **header.php**.

Puis coupez depuis la balise fermant la zone de contenu jusque le bas de la page, pour mettre tout ça dans **footer.php**

Enfin, coupez la barre latérale (le contenu des balises **<aside>**) et placez ce code dans son propre fichier **sidebar.php.**

Organiser les fichiers de base du thème

Appelez vos fichiers nouvellement crées dans index.php

Dans **index.php**, ajoutez les fonctions **<?php get_header(); ?>** pour appeler le fichier **header.php** au bon endroit.
Faites de même pour **footer.php**,avec **<?php get_footer(); ?>**
Idem pour la barre latérale. Appelez **sidebar.php** à l'aide de **<?php get_sidebar(); ?>**

Créez un fichier functions.php

Le fichier **functions.php** est essentiel: il va contenir toutes les fonctionnalités du thème et les déclarations de support pour les fonctionnalités natives de WordPress: logo, image d'entête personnalisée, images à la une, etc...

Charger styles et scripts correctement

Hookez sur wp_enqueue_scripts pour charger vos ressources

Dans **functions.php**, créez une fonction et utilisez **<?php wp_enqueue_style(); ?>** pour charger vos fichiers .css et et **<?php wp_enqueue_script(); ?>** pour les fichiers .js. Hookez cette fonction sur **wp_enqueue_scripts**.

Si vous utilisez une Google Font, utilisez également **<?php wp_enqueue_style(); ?>** pour charger la police.

Insérez les hooks nécessaires dans header.php et footer.php

Dans **header.php**, ajoutez la fonction **<?php wp_head(); ?>** dans la balise **<head>**. Cela va placer un hook pour permettre à WordPress et autres plugins d'écrire dans cette balise.

De la même façon, la fonction **<?php wp_footer(); ?>** est à placer juste avant la balise fermante **</body>**, dans **footer.php** et permet d'écrire les scripts à cet endroit.

Mettre en place la navigation

Déclarez une zone de menu dans l'administration.

Dans **functions.php**, créez une fonction et utilisez **<?php register_nav_menus(); ?>** pour déclarer une zone de menu dans l'administration de WordPress.
Hookez cette fonction sur **after_setup_theme**.

Affichez votre menu sur le devant du site

Dans **header.php**, utilisez **<?php wp_nav_menu(): ?>** pour afficher le menu. Utilisez le paramètre **theme_location** pour définir la zone de menu à afficher.

Mettre en place la boucle

Placez la boucle de WordPress dans index.php

Dans **index.php**, utilisez les fonctions **<?php have_posts(); ?>** et **<?php the_post(); ?>** pour mettre en place la boucle qui va permettre d'afficher le contenu de chaque page.

Dans la boucle, utilisez **<?php get_template_part(); ?>** pour appeler le fichier responsable de l'affichage de l'article.

Coupez-collez le balisage d'un article dans son propre fichier

Créez un dossier **template-parts/**, puis créez-y un fichier **content.php.**
Collez-y le balisage utilisé pour un article simple.

Mettre en place la zone de widget principale

Déclarez une zone de widget dans functions.php

Dans **functions.php**, toujours dans la fonction hookée sur **after_setup_theme**, utilisez la fonction **<?php register_sidebar(); ?>** pour déclarer une zone de widget administrable.

Attention à garder **%1$s** et **%2$s** dans le paramètre '**before_widget**', pour permettre à WordPress de générer des classes dynamiquement pour vos widgets.

Affichez votre zone de widget dans sidebar.php

Dans **sidebar.php**, utilisez la fonction **<?php is_active_sidebar(); ?>** pour vérifier qu'il y a bien des widgets à afficher, puis la fonction **<?php dynamic_sidebar(); ?>** pour afficher les widgets enregistrés pour la zone passée en paramètre.

Remplacer notre contenu codé en dur

post_class et body_class

<?php post_class();)> est une fonction à ajouter sur la balise **<article>** qui va permettre de générer toute une série de classes bien utiles.

De la même manière, **<?php body_class(); ?>** sur la balise **<body>** va permettre d'ajouter des classes CSS très utiles en fonction de la page demandée.

Remplacez le contenu codé en dur par des templates tags

Dans **content.php**, remplacez les éléments de contenu codés en dur dans le prototype par les templates tags de WordPress correspondants: **<?php the title(); ?>**, **<?php the_content(); ?>**, **<?php the_date(); ?>**, etc...

Cela peut être utile aussi d'ajouter l'identifiant de l'article sur la balise **<article>** à l'aide de **<?php the_ID(); ?>**

Remplacer notre contenu codé en dur

Remplacez le contenu codé en dur dans header.php

Même travail dans **header.php**. Utilisez **<?php bloginfo(); ?>** pour afficher les données sur le site. N'oubliez pas d'utiliser **<?php language_attribute(); ?>** sur la balise **<html>**, et d'ajouter **<?php body_class(); ?>** sur la balise **<body>** si ce n'est pas déjà fait.

Vous pouvez enlever la balise **<title>** et laisser WordPress s'occuper des titres en ajoutant **<?php add_theme_support('title-tag'); ?>** dans **functions.php**.

Même punition pour footer.php

Remplacez le contenu codé en dur par le contenu dynamique correspondant.

Gardez le code de vos fichiers de template clean

Ne dupliquez pas de code !

Isolez vos template tags personnalisés dans leur propre fichier, que vous allez inclure dans **functions.php**. Le but est de ne pas surcharger ce fichier.

Créez aussi un fichier qui va contenir vos autres fonctions qui ne sont pas des templates tags, c'est-à-dire qui ne servent pas à afficher du contenu.

Un thème mono-template

A ce stade, votre thème devrait être fonctionnel. Félicitations !

Vous avez un unique template qui va servir tout votre contenu, mais vous avez une bonne fondation sur laquelle bâtir vos autres templates !

Créez les autres templates

Créez un template single.php

single.php est le template utilisé pour les pages d'article simple. Il est fortement conseillé de l'inclure. Si WordPress ne trouve pas **single.php**, il utilisera **index.php** pour les articles.

N'oubliez pas d'inclure **<?php comments_template(); ?>** pour afficher la zone de commentaires.

Créez un template page.php

page.php contrôle comment les pages simples du thème vont s'afficher. Comme pour **single.php**, il est fortement recommandé de l'inclure. Si WordPress ne trouve pas **page.php**, il utilisera **index.php** pour les pages.

Si le design de vos pages et de vos articles est très proche, vous pouvez vous en sortir avec **singular.php**, au lieu de **page.php** + **single.php**.

Créez les autres templates

Créez un template archive.php

archive.php est le template utilisé pour les pages d'archives du site : par catégorie, par étiquette, par auteur, par date, etc...

Il va surement ressembler à **index.php**, mais n'oubliez pas d'inclure un titre et une description avec **<?php the_archive_title(); ?>** et **<?php the_archive_description(); ?>**

Créez un template search.php

search.php est utilisé pour afficher les résultats de recherche.
En général, utiliser **<?php the_excerpt(); ?>** pour afficher uniquement l'extrait des contenus trouvés est suffisant.

Créez les autres templates

Créez un template 404.php

404.php est utilisé pour les pages d'erreurs 404. Un message d'erreur et un formulaire de recherche ou d'autres liens utiles peuvent suffire, mais vous verrez parfois des pages très originales sur certains sites !

Créez un template home.php

home.php est le template de la liste de vos articles, c'est-à-dire votre blog.
Il ressemblera surement à **index.php**, mais dans le cas de ce template, vous êtes sûr d'avoir affaire à une liste d'article, donc il pourra contenir moins de logique.

Ajoutez les fonctionnalités courantes attendues

Un logo personnalisé

Cette fonctionnalité s'active avec **<?php add_theme_support('custom-logo'); ?>**
Vous pouvez passer un tableau de paramètre pour plus de contrôle.
Le logo s'affiche à l'aide de la fonction **<?php the_custom_logo(); ?>**

Une image d'entête personnalisée

Cette fonctionnalité s'active avec **<?php add_theme_support('custom-header'); ?>**
Comme pour le logo, vous pouvez passer un tableau de paramètre pour plus de contrôle.
L'url de l'image s'obtient avec **<?php header_image(); ?>**. Vous pouvez ensuite l'afficher dans une balise **<img>.**

Ajoutez les fonctionnalités courantes attendues

Un background personnalisé

Un simple **<?php add_theme_support('custom-background'); ?>** permet d'activer l'option dans l'outil de personnalisation. Idem, vous pouvez passer un tableau de paramètre pour plus de contrôle.

Balisage HTML5 pour les éléments générés par WordPress

Pour que les commentaires soient balisés en HTML5, il faut ajouter **<?php add_theme_support('html5', array('search-form', 'comment-form', 'comment-list', 'gallery', 'caption')); ?>**

Vers l'infini et au-delà

L'outil de personnalisation de WordPress

Une fois les templates de base mis en place et les fonctionnalités courantes implémentées, vous pouvez ajouter des réglages dans l'outil de personnalisation, des templates de pages personnalisés ou d'autres fonctionnalités.

Les extensions

Améliorer la compatibilité avec certaines extensions est aussi une possibilité : Jetpack, Woocommerce, Easy Digital Download, RevSlider, ...
Vous avez le choix !

Sky is the limit !

Merci !

Merci !

J'espère que ça vous a plu et que ça vous a été utile !
Si vous avez des questions, n'hésitez pas !

vincedubroeucq sur Twitter
https://vincentdubroeucq.com