Hooks

Un **hook WordPress**, peut-être de deux types : « **action** » ou « **filtre**« , il s'agit d'un mécanisme intégré dans le noyau de WordPress qui **permet aux développeurs d'interagir avec le système et d'étendre ou de modifier son fonctionnement sans modifier directement le code principal.**

Les hooks sont essentiels pour personnaliser et étendre les fonctionnalités de WordPress de manière modulaire et robuste.

Il existe deux types de hooks dans WordPress : les **actions** (actions hooks) et les **filtres** (filter hooks).

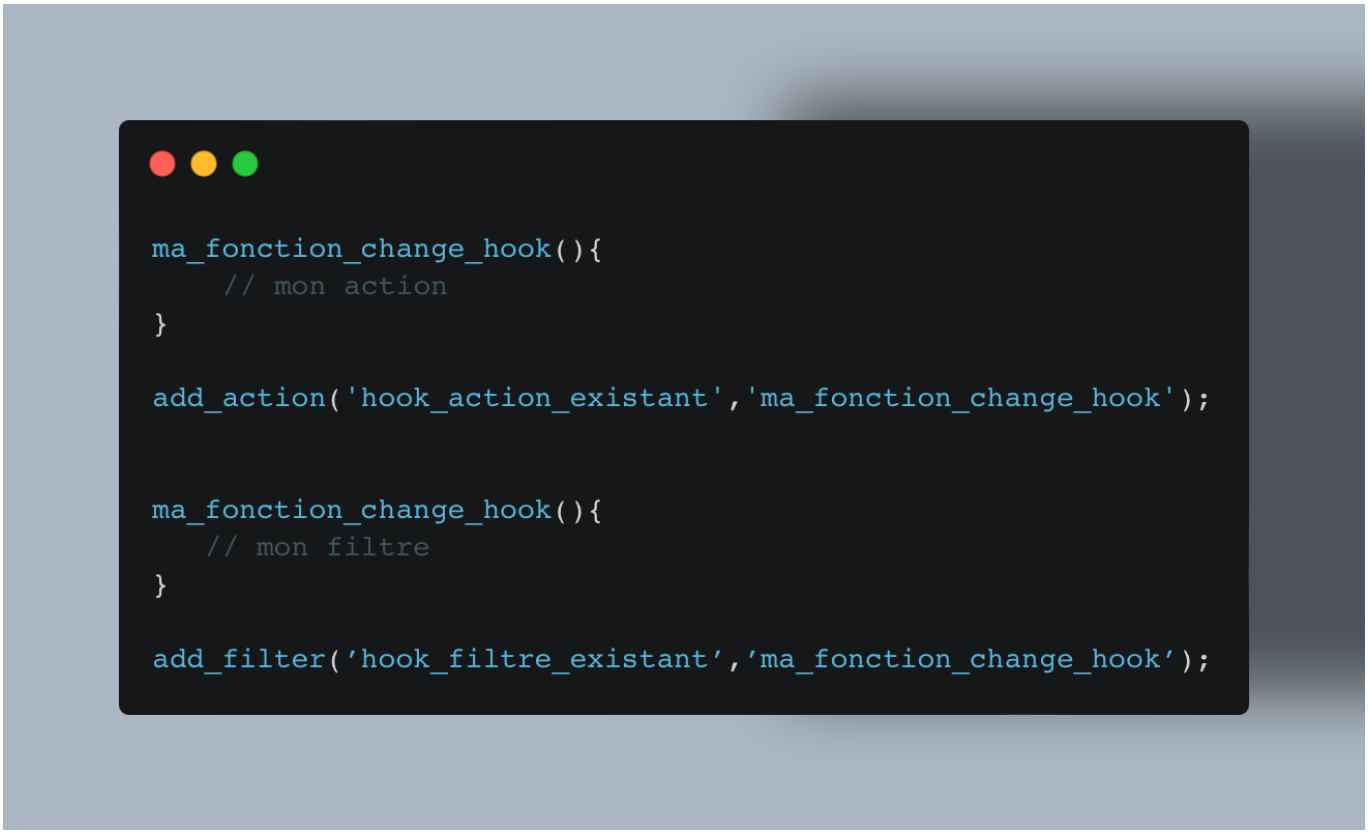
- **Action hooks :** permettent aux développeurs d'**exécuter leur propre code à des points spécifiques de l'exécution de WordPress.** Ils agissent comme des **points d'ancrage où vous pouvez attacher vos propres fonctions personnalisées**. Par exemple, l'action « **init** » est déclenchée au démarrage de WordPress, et vous pouvez y attacher une fonction personnalisée pour effectuer certaines tâches d'initialisation.
- **Filter hooks : modifier les données avant qu'elles ne soient affichées ou enregistrées dans la base de données**. Ils fournissent **une méthode pour filtrer ou modifier les données de WordPress avant qu'elles ne soient utilisées**. Par exemple, le filtre « **the_content** » permet de modifier le contenu d'un article avant qu'il ne soit affiché à l'utilisateur.

Les **hooks** sont utilisés dans les thèmes WordPress et les plugins pour ajouter de nouvelles fonctionnalités, modifier des fonctionnalités existantes, manipuler des données et exécuter des actions personnalisées à des moments spécifiques du flux d'exécution de WordPress.

Nativement + de 2300 hooks dans le cœur du CMS WordPress.

Pour utiliser un **hook**, vous devez attacher une fonction personnalisée à ce hook à l'aide de fonctions spécifiques fournies par WordPress, telles que « **add_action** » pour les **actions hooks** et « **add_filter** » pour les **filtres hooks**.

Ces fonctions prennent en paramètre le **nom du hook** et le **nom de la fonction à exécuter**.



```php
ma_fonction_change_hook(){
    // mon action
}

add_action('hook_action_existant','ma_fonction_change_hook');


ma_fonction_change_hook(){
   // mon filtre
}

add_filter('hook_filtre_existant','ma_fonction_change_hook');
```

L'ordre d'exécution des hooks du front

Hook	Description
mu_plugin_loaded	Après chargement des mu-plugins
plugins_loaded	Après que les extensions soient chargées
after_setup_theme	Après que le fichier **functions.php** du thème soit chargé Beaucoup de fonctionnalités du thème sont déclarées à ce moment-là à l'aide de **add_theme_support()**
init	WordPress est en grande partie chargé (thème et extensions, l'utilisateur est authentifié et disponible)Utilisé pour déclarer les CPT, taxonomies, …
wp_loaded	WordPress est complètement chargé, mais la requête n'a pas encore été traitée
parse_request	Les variables de la requête HTTP entrante sont interprétées
send_headers	Les entêtes HTTP sont envoyées
parse_query	Les variables pour la requête principale **WP_Query** sont prêtesLa requête pour aller chercher le bon contenu n'est pas encore faite
pre_get_posts	Hook permettant de manipuler la requête **WP_Query** avant qu'elle ne soit effectuée
wp	L'objet **$wp** est complet
template_redirect	Exécuté avant que WordPress ne cherche le bon modèle de page dans le thème pour afficher le contenu
get_header	Juste avant de charger **header.php**

Hook	Description
wp_enqueue_styleswp_enqueue_scripts	Hook sur lequel se greffer pour ajouter des styles et scripts à charger
wp_head	Utiliser ce hook pour écrire dans la balise **<head>** de la page
wp_print_styleswp_print_scripts	Se déclenchent juste avant d'afficher les balises **<link>** et **<script>** dans le **<head>**
the_post	Se déclenche quand l'objet $post est prêt
pre_get_comments	Déclenché juste avant d'aller chercher les commentaires
get_sidebar	Avant que le modèle pour la zone de widget soit chargée
get_footer	Avant d'aller chercher **footer.php**
wp_footer	Déclenché dans le pied de page pour charger les balises <script> des extensions
admin_bar_menu	Charge le menu de la barre d'administration
shutdown	Juste avant que l'exécution de PHP s'arrête

Astuces : les hooks peuvent être appelés directement depuis votre template ….

- **hook d'action** grâce à la fonction **do_action('nomdevotrehook')**
- **hook de filtre** grâce à la fonction **apply_filters('nomdevotrehook')**

Hooks : paramètres et exemples

Les paramètres

- **étiquette :** nom de l'action à cibler
- **callback :** fonction appelée
- **priorité :** permet de gérer l'ordre d'exécution dans le cas où il y aurait plusieurs fonctions greffées sur ce hook — priorité inférieure = action appelée avant (par défaut à 10)

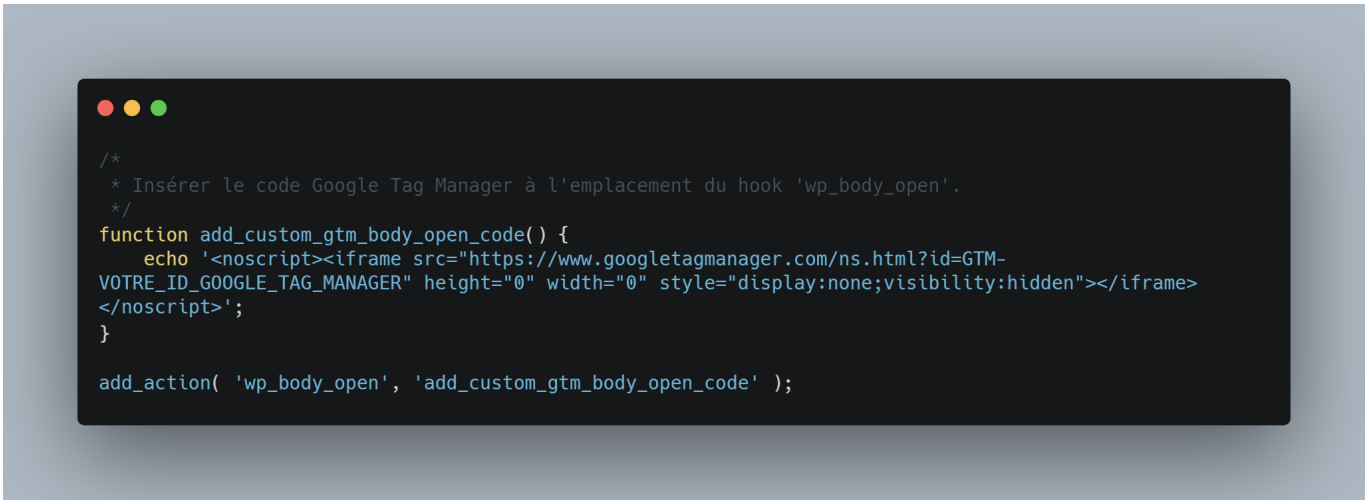
Exemples

admin_menu

```php
<?php
  /* Supprimer des éléments de menus du Tableau de bord WordPress */
function custom_remove_menu_pages() {
    remove_menu_page( 'tools.php' );
    remove_menu_page( 'edit-comments.php' );
}
add_action( 'admin_menu', 'custom_remove_menu_pages' );
```

Ex. supprimer des menus du Tableau de bord WordPress via la fonction **remove_menu_page()**

wp_body_open



```php
/*
 * Insérer le code Google Tag Manager à l'emplacement du hook 'wp_body_open'.
 */
function add_custom_gtm_body_open_code() {
    echo '<noscript><iframe src="https://www.googletagmanager.com/ns.html?id=GTM-
VOTRE_ID_GOOGLE_TAG_MANAGER" height="0" width="0" style="display:none;visibility:hidden"></iframe>
</noscript>';
}

add_action( 'wp_body_open', 'add_custom_gtm_body_open_code' );
```

Ex. insérer le code de suivi **Google Tag Manager**

wp_head

```php
<?php
  /**
   * Ajouter le Tag Google Analitycs dans le Head
   *
   * @link https://codex.wordpress.org/Plugin_API/Action_Reference/wp_head
   * @link
https://developers.google.com/analytics/devguides/collection/gtagjs/
   */
  add_action( 'wp_head', function () {
      // Ne suit pas les admins (on peut faire pareil pour le RGPD ici)
      if ( !current_user_can( 'edit_pages' ) ) {
          echo "
              <!-- Global site tag (gtag.js) - Google Analytics -->
              <script async src=\"https://www.googletagmanager.com/gtag/js?
id=GA_TRACKING_ID\"></script>
              <script>
                window.dataLayer = window.dataLayer || [];
                function gtag(){dataLayer.push(arguments);}
                gtag('js', new Date());

                gtag('config', 'GA_TRACKING_ID');
              </script>
          ";
      }
  }, 0 );
```

Ex. ajouter le code de suivi **Google Analytics**

pre_get_posts

```php
<?php

    function search_filter($query) {
      if ( !is_admin() && $query->is_main_query() ) {
        if ($query->is_search) {
          // Ajouter une recherche dans les Custom Post Type = ex. Recettes
          $query->set('post_type', array('page', 'post', 'recipe' ) );
        }
      }
    }

    add_action('pre_get_posts','search_filter');
```

Ex. modifier le retour d’une recherche, en ajoutant la possibilité de chercher dans un CPT (ici ‘recipe’ pour des recettes de cuisine)

body_class

```php
function add_your_body_class($classes) {
    $classes[] = 'new-class';
    return $classes;
}

add_filter('body_class', 'add_your_body_class');
```

Ex. ajouter une classe CSS dans le body

save_post

Le hook **`save_post`** est un hook d'action de base de WordPress qui est **déclenché chaque fois qu'un article ou une page est enregistré dans la base de données**.

Vous pouvez utiliser ce hook pour effectuer des actions spécifiques lorsque le contenu est sauvegardé, modifié ou publié.

```
add_action('save_post', 'mon_action_apres_enregistrement', 10, 3);

function mon_action_apres_enregistrement($post_id, $post, $update) {
  // votre code ici pour effectuer des actions après l'enregistrement du
contenu

  // pour éviter d'exécuter le code en cas d'enregistrement d'une révision
  if( wp_is_post_revision( $post_id ) ) { return; }

  // pour éviter d'exécuter le code en cas de sauvegardes automatiques
  if( defined( 'DOING_AUTOSAVE' ) and DOING_AUTOSAVE ) { return; }

  // vérifier si l'action est un enregistrement initial ou une mise à jour
  if ($update) {
    // c'est une mise à jour
  } else {
    // c'est un nouvel enregistrement
    // définir les termes de taxonomie personnalisée pour le nouvel article
ici
    $term_ids = array(1, 2, 3); // remplacez par les ID des termes de
```

```
taxonomie souhaités
    $taxonomy = 'ma_taxonomie'; // remplacez par le slug de votre taxonomie
personnalisée ou 'category'
    wp_set_post_terms($post_id, $term_ids, $taxonomy);
  }

  // récupérer le type de contenu (post_type)
  $post_type = get_post_type($post_id);

  // faire des actions spécifiques en fonction du type de contenu
  if ($post_type === 'post') {
    // actions spécifiques aux articles
  } elseif ($post_type === 'page') {
    // actions spécifiques aux pages
  }

  // récupérer les données du formulaire et les traiter
  if (isset($_POST['mon_champ'])) {
    $valeur_champ = sanitize_text_field($_POST['mon_champ']);
    // faire quelque chose avec $valeur_champ
  }
}
```

Le hook `save_post` reçoit trois paramètres :

- **`$post_id` :** l'ID de l'article ou de la page qui est enregistré.
- **`$post` :** l'objet post contenant les données de l'article ou de la page.
- **`$update` :** un booléen indiquant s'il s'agit d'une mise à jour (**`true`**) ou d'un nouvel enregistrement (**`false`**).

Rappel : les révisions sont des sauvegardes automatiques créées par WordPress chaque fois qu'un article ou une page est modifié et sauvegardé.

Remarque : l'**autosave** (sauvegarde automatique) est une fonctionnalité WordPress qui permet de sauvegarder automatiquement les contenus en cours d'édition, à intervalles réguliers pendant que vous les rédigez.

Désactiver un hook

Pour désactiver un hook WordPress, vous pouvez utiliser la fonction **`remove_action()`** pour les **actions hooks** ou **`remove_filter()`** pour les **filtres hooks**.

Si la fonction a été attachée à l'action hook avec une priorité spécifique (différente de 10 qui est la valeur par défaut), indiquez la même priorité utilisée lors de l'ajout de la fonction à l'action hook. Sinon, vous pouvez omettre ce paramètre.

On stoppe l'exécution d'une fonction préalablement hookée.

La fonction en paramètre ne s'exécutera pas, elle est retirée de la liste d'attente.

Assurez-vous d'appeler les fonctions **`remove_action()`** et **`remove_filter()`** dès que le hook que vous souhaitez désactiver a été ajouté, par exemple dans le fichier **`functions.php`** de votre thème ou dans un plugin.