# WordPress : créer ses propres shortcodes

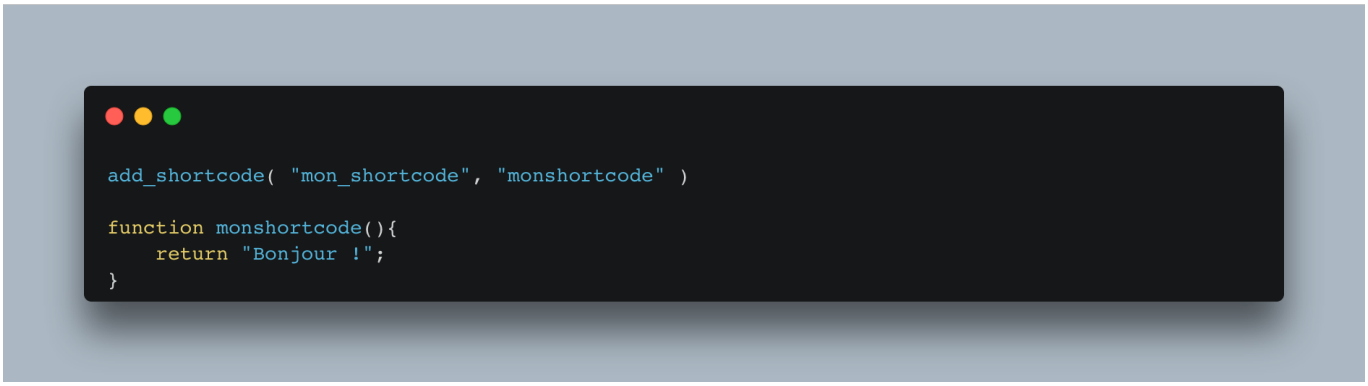
## Le principe des shortcodes

Un **shortcode** (**code court**) permet d'injecter du code (**PHP**, **HTML**, **JS** et **CSS**) à n'importe quel endroit du site WordPress.

**Format d'affichage : `[mon_shortcode]`** ou avec paramètres **`[mon_shortcode name="Mickaël"]`**

## Créer ses premiers shortcodes réguliers

Vous pouvez jouter un appel à **`add_shortcode()`** depuis le fichier **functions.php** de votre thème ou depuis un plugin (méthode plus propre).




**Remarque :** il ne faut pas afficher une valeur dans le shortcode, mais la retourner à la fin.

**Version avec l'utilisation de la fonction `wp_get_current_user()` pour retourner des informations sur l'utilisateur actif :**

```php
<?php
function monshortcode( $args ) {

    // L'utilisateur n'est pas connecté
    if( ! is_user_logged_in() ) {
        return 'invité';
    }

    // Retrouver l'utilisateur actuel
    $current_user = wp_get_current_user();

    // Retourner son prénom
    return "Bonjour " . $current_user->user_firstname . " !";
}

add_shortcode( 'mon_shortcode', 'monshortcode' );
```

**Un exemple avec un shortcode paramétrable :**

```php
add_shortcode( "mon_shortcode", "monshortcode" )

function monshortcode($args){

    $name = $args["name"];

    return "Bonjour ".$name." !";
}
```

**Version avec l'utilisation de la fonction `get_userdata()` pour retourner des informations sur l'utilisateur actif :**

```php
<?php
function monshortcode( $args ) {
    // Combiner les attributs transmis avec des valeurs par défaut si nécessaire.
    $args = shortcode_atts( array(
        // Valeur par défaut : l'identifiant de l'utilisateur actif
        'id' => get_current_user_id(),
    ), $args, 'mon_shortcode' );

    // L'utilisateur dont l'ID a été transmis n'est pas connecté
    if( ! $args['id'] ) {
        return 'invité';
    }

    // Retrouver l'utilisateur dont l'ID a été transmis
    $user = get_userdata( $args['id'] );

    // Retourner son prénom
    return "Bonjour " . $user->user_firstname . " !";
}

add_shortcode( 'mon_shortcode', 'monshortcode' );
```

## Les shortcodes englobants

Un **shortcode WordPress englobant (wrap)** est une fonctionnalité qui vous permet de créer **un shortcode personnalisé dans WordPress qui englobe le contenu existant**.

Par exemple, vous pouvez créer un shortcode englobant **`[encadrer]`** qui ajoute un cadre stylisé autour du contenu existant, sans le modifier.

Lorsque vous utiliserez ce shortcode dans votre contenu, il encadrera le contenu sans le modifier.

```php
<?php

function shortcode_encadrer( $args, $content ) {

    return '<div class="cadre">' . $content . '</div>' ;
}

add_shortcode( 'encadrer', 'shortcode_encadrer' );
```

## Afficher le rendu d'un shortcode

La fonction **`the_content()`** affiche le rendu de vos shortcodes WordPress, à condition qu'ils soient intégrés dans le contenu principal de la publication.

Si vous souhaitez afficher le rendu d'un shortcode dans un autre endroit spécifique, utilisez la fonction `do_shortcode()` ou sa nouvelle version `apply_shortcode()`.

Cette fonction va exécuter le shortcode WordPress et récupérer son contenu.

```
$content = "Voici du contenu avec un shortcode : [mon_shortcode]";

$shortcode_content = apply_shortcode($content);

echo $shortcode_content;
```

Dans cet exemple, nous avons une variable **`$content`** qui contient du texte avec un shortcode **`[mon_shortcode]`**.

**Remarque :** assurez-vous de remplacer **`[mon_shortcode]`** par le nom du shortcode que vous souhaitez exécuter.

En utilisant **`apply_shortcode()`**, nous exécutons le shortcode et récupérons son contenu dans la variable **`$shortcode_content`**.

Ensuite, nous l'affichons en utilisant **`echo`**.

## Shortcodes et WP Query

**WP Query** est une classe de WordPress qui permet de récupérer des contenus de la base de données selon des critères spécifiques.

**Elle est utilisée pour exécuter des requêtes personnalisées** sur les articles, les pages, les types de contenu personnalisés et autres éléments de WordPress.

Avec WP Query, vous pouvez spécifier divers paramètres pour filtrer et trier les résultats de la requête.

Une fois que vous avez instancié un objet WP Query avec vos paramètres, vous pouvez parcourir les résultats à l'aide d'une boucle **`while`** et accéder aux informations de chaque élément récupéré.

**Remarque :** n'oubliez pas d'appeler **`wp_reset_postdata()`** après avoir terminé votre boucle WP Query pour réinitialiser les données de publication.

**Exemple complet avec récupération des valeurs d'un CPT :**

```php
<?php

    // Créer un shortcode pour afficher les compétences
    function reg_display_competences()
    {
        $args = array(
            'post_type' => 'competence', // Nom du Custom Post Type
            'post_status' => 'publish', // Status
            'posts_per_page' => 5, // Nombre maximum d'élément
            'order' => 'DESC', // Du plus grand au plus petit
            'orderby' => 'ID' // Ordonner par l'ID
        );

        $s = '';

        $query = new WP_Query($args);

        if ($query->have_posts()) {

            $s .= '<div class="competences">';

            $nombreCompetences = $query->found_posts;

            echo '<div class="card-header">' . __('Number of capabilities : ', LANGUAGE_ZONE) . $nombreCompetences . '</div>';
            $s .= '<ul>';
            $s .= '<div class="card-body">';

            while ($query->have_posts()) {

                $query->the_post();

                $s .= '<li>';
                $s .= '<a href="' . get_the_permalink() . '">' . get_the_title() . '</a>';
                $s .= '</li>';
            }

            $s .= '</div>';
            $s .= '</ul>';
            $s .= '</div>';
         }

         wp_reset_postdata();
         return $s;

    }

    add_shortcode('competences', 'reg_display_competences');
```

**Conseil :** pour améliorer la gestion de votre code source, vous pouvez utiliser la fonction **`get_template_part()`** de WordPress qui permet d'inclure des fichiers de modèles réutilisables dans un thème.