WordPress : charger scripts JavaScript et styles CSS

Pour charger des scripts et des styles dans le fichier **functions.php** d'un thème WordPress, vous pouvez utiliser les fonctions suivantes qui permettent de bien gérer les **dépendances**.

Exemple : si vous souhaitez charger la bibliothèque JavaScript **select2**, vous aurez besoin que **jQuery** soit chargé au préalable.

Évitez de charger vos scripts directement dans le **header.php** du thème.

Le **hook** « **wp_enqueue_{style|script}** » de WordPress permet de charger les styles et scripts du thème.

Assurez-vous de modifier les chemins des fichiers JavaScript et CSS en fonction de l'emplacement réel de vos fichiers dans votre thème.

Le principe : le thème et les extensions déclarent leurs scripts et styles à WordPress, spécifiant les dépendances requises pour chacun, WordPress gère alors automatiquement le chargement dans l'ordre approprié.

Pour charger des scripts

```
function enqueue_custom_scripts() {
    // Enqueue your custom script
    wp_enqueue_script('custom-script', get_template_directory_uri() .
'/js/custom-script.js', array('jquery'), '1.0', true);
}

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

Dans cet exemple, nous utilisons la fonction **`wp_enqueue_script()`** pour charger un script personnalisé appelé « custom-script.js » (on parle de **handle** pour ce nom de fichier).

Vous devez spécifier le chemin vers votre fichier JavaScript et lui donner un **identifiant unique** (ex. « custom-script.js »).

La fonction **`get_template_directory_uri()`** permet d'obtenir l'emplacement du thème.

Attention ! Si vous disposez d'un **thème enfant**, il vous faudra utiliser la fonction **`get_stylesheet_directory_uri()`** au lieu de **`get_template_directory_uri()`**.

Vous pouvez également définir des **dépendances** en déclarant leur **handle** dans un **tableau = array** (dans cet exemple, nous avons ajouté « **jquery** » comme dépendance) et spécifier la **version du script**.

Le numéro de version apparaît dans les paramètres de l'URL lors de la déclaration du chargement effectif des scripts ou styles.

Ex. custom-script.js?ver=1.0

Enfin, en définissant le dernier argument sur **`true`**, le script sera chargé dans le **pied de page** via **wp_footer** (**`false`** pour le charger dans l'**en-tête** via **wp_header**).

Conseil : le chargement des scripts en pied de page va vous permettre d'améliorer les performances.

Les scripts sont en général exécutés que lorsqu'un événement **Dom ready** est déclenché, c'est-à-dire uniquement au moment où la totalité de la page web est chargée et prête à être manipulée. Il est donc inutile de charger les scripts trop tôt.

Le numéro de version indiqué dans **wp_enqueue_script()** et **wp_enqueue_style()** est à changer pour vider le cache (⇔ forcer la prise en compte de la nouvelle version).

Cela vise à résoudre un problème classique : les navigateurs web mettent en cache certains fichiers pour éviter de les charger à chaque visite sur votre site.

Lorsque ces fichiers sont modifiés sur le serveur, les changements ne sont pas immédiatement pris en compte.

Cela peut entraîner des différences d'affichage sur votre site web.

La mise en place du numéro de version et sa modification permet de contourner cette problématique.

Pour charger des styles

```
function enqueue_custom_styles() {
    // Enqueue your custom styles
    wp_enqueue_style('custom-style', get_template_directory_uri() .
'/css/custom-style.css', array(), '1.0', 'all');
}

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

Dans cet exemple, nous utilisons la fonction **`wp_enqueue_style()`** pour charger un style personnalisé appelé « custom-style.css ».

Vous devez spécifier le chemin vers votre fichier CSS, lui donner un identifiant unique et spécifier la version du style.

L'argument **`'all'`** spécifie que le style peut être utilisé pour tous les médias (smartphone, tablette et ordinateur).

À noter : lorsque vous avez un thème enfant, le chargement de la feuille de style CSS style.css du thème parent n'est pas automatique, il vous faut la charger manuellement.

```
<?php

function montheme_child_load_styles() {
    // Chargement de la feuille du style du thème parent
    wp_enqueue_style( 'montheme-parent-theme', get_template_directory_uri() .
'/style.css' );
```

```
    // Chargement de la feuille de style du thème enfant
    wp_enqueue_style( 'montheme-child-theme', get_stylesheet_directory_uri()
. '/style.css' );
}

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

Pour rappel … :

- **get_template_directory_uri() :** pour des fichiers dans le **thème parent**
- **get_stylesheet_directory_uri() :** pour des fichiers dans le **thème enfant**

Cas pratique : chargement des composants Bootstrap

```
<?php

    /* Chargement de Bootstrap */
    function montheme_load_bootstrap(){
        wp_enqueue_style('style', get_stylesheet_uri());

        wp_enqueue_style('bootstrap',
'https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css');
        wp_enqueue_script('jquery');

        wp_enqueue_script('bootstrap',
'https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js',
array('jquery'));
    }

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

À partir de ce moment, vous pouvez commencer à construire votre thème via **Bootstrap**.

Ex. header.php

```php
<html>

<head>
    <meta charset="<?php bloginfo('charset'); ?>">
    <title><?php wp_title('|', true, 'right'); ?></title>
    <!--[if lt IE 9]>
    <script src="<?php echo get_template_directory_uri(); ?>/js/html5.js"></script>
    <![endif]-->
    <?php wp_head(); ?>
</head>

<body <?php body_class(); ?>>
    <div id="page" class="hfeed site">
        <header id="masthead">

            <h1 class="site-title"><?php bloginfo('name'); ?></h1>
            <h2 class="site-description"><?php bloginfo('description'); ?></h2>

            <div id="navbar" class="navbar">
                <nav id="site-navigation" class="navigation main-navigation" role="navigation">
                    <?php
                    wp_nav_menu(
                        array(
                            'theme_location' => 'primary',
                            'menu_class' => 'nav-menu'
                        )
                    );
                    ?>
                    <?php get_search_form(); ?>
                </nav><!-- #site-navigation -->
            </div><!-- #navbar -->
        </header><!-- #masthead -->
        <div id="main" class="site-main">
```

Chargement conditionnel des scripts et styles

Pour charger des scripts et des styles WordPress en utilisant des **balises conditionnelles (conditional tags)**, vous pouvez utiliser les fonctions **`wp_enqueue_script()`** et **`wp_enqueue_style()`** avec des conditions spécifiques.

Les balises conditionnelles vous permettent de vérifier certaines conditions, telles que la page en cours, le type de publication, le rôle de l'utilisateur, etc., pour décider quels scripts ou styles doivent être chargés.

Voici un exemple :

```
function enqueue_custom_scripts_styles() {
    // Charger le script uniquement sur la page d'accueil
    if (is_front_page()) {
        wp_enqueue_script('custom-script', get_template_directory_uri() .
'/js/custom-script.js', array('jquery'), '1.0', true);
    }

    // Charger le style uniquement sur les pages de publication (articles)
    if (is_single()) {
```

```
        wp_enqueue_style('custom-style', get_template_directory_uri() .
'/css/custom-style.css', array(), '1.0', 'all');
    }
}

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

Dans cet exemple, nous utilisons deux balises conditionnelles :

- **`is_front_page()`** : cette balise vérifie si la page en cours est la page d'accueil.
- **`is_single()` :** cette balise vérifie si la page en cours est une page de publication individuelle (un article).

Cela vous permet de charger les scripts et les styles uniquement sur les pages spécifiques où ils sont nécessaires, ce qui peut contribuer à optimiser les performances de votre site WordPress.