



Formation TP WEB DESIGNER

Réaliser un thème sous WORDPRESS

Table des matières

PLAN DE TP ET OBJECTIF PÉDAGOGIQUE	5
1- Objectif de réalisation.....	5
2-Plan de TP.....	5
RÉALISATION DU THÈME	6
1-Initialisation.....	6
A) MISE EN PLACE DU DOSSIER DU THÈME.....	6
B) DANS LE FICHIER STYLE.CSS.....	6
C) ACTIVER LE THÈME.....	6
2-Le fichier index.php.....	7
3-La boucle.....	8
4-Les données de l'article.....	8
A) VOIR EN LIGNE.....	8
5-Activer des fonctionnalités : le fichier de fonctions.....	9
A) PERMETTRE DE PERSONNALISER LE THÈME DEPUIS L'ADMINISTRATION : ADD_THEME_SUPPORT.....	9
➔ Image d'en-tête.....	9
➔ Personnaliser le logo du site.....	10
➔ Balisage html5 sur les contenus html générés par les fonctions.....	11
➔ Image à la une sur les posts.....	11
B) PERMETTRE DE PERSONNALISER DES MENUS DEPUIS L'ADMINISTRATION : REGISTER_NAV_MENUS().....	12
C) CRÉER UNE FONCTION POUR RAJOUTER UN LIEN VERS L'ACCUEIL AUTOMATIQUEMENT.....	13
D) PRÉVOIR L'AJOUT DE WIDGETS DANS UNE SIDEBAR : REGISTER_SIDEBAR().....	13
6-Permettre à des plugins des déclarations supplémentaires dans le code html.....	14
➔ Observer l'apparition d'une barre de navigation en haut de la page :	14
7-Découper le template.....	15
A) CRÉER LES FICHIERS SUIVANTS À LA RACINE DU THÈME :	15
B) CRÉER UN DOSSIER TEMPLATES-PART À LA RACINE DU THÈME.....	15
C) CRÉER LES FICHIER SUIVANTS À L'INTÉRIEUR DU DOSSIER TEMPLATES-PART.....	15
D) DÉCOUPER LE TEMPLATE.....	15
E) INCLURE LES PARTIES EXTRAITES DANS LE TEMPLATE.....	16
➔ Header,footer,sidebar.....	16
➔ template-parts.....	16
F) LES POST FORMATS.....	17
➔ Activer les posts formats pour le thème.....	18
➔ paramétrer un post format sur un article.....	18
➔ Un template-part par post-format.....	19
8-Les templates à réaliser.....	20
A) INDEX.PHP.....	20
B) ARCHIVE.PHP.....	20
C) SINGLE.PHP.....	20
D) PAGE.PHP.....	20
9-La hiérarchie des templates.....	20
10-Les commentaires.....	21
A) LE TEMPLATE COMMENTS.PHP.....	21
B) GESTION DES COMMENTAIRES.....	21
C) LES RÉTROLIENS.....	21
11-Le moteur de recherche.....	21
12-Les médias sous WORDPRESS.....	22
A) INSTALLATION DU PLUGIN SIMPLE IMAGE SIZES.....	22
B) CRÉER DE NOUVEAUX FORMATS.....	22

C) AFFICHAGE D'UN VISUEL EN FONCTION D'UN FORMAT.....	22
D) INSÉRER UNE GALERIE DANS UN ARTICLE.....	23
E) INSTALLER UN PLUGIN LIGHTBOX.....	32
13-Insertion de css et de scripts via le fichier de fonctions.....	33
A) INSÉRER DES FEUILLES DE STYLES.....	33
B) INSÉRER DES SCRIPTS.....	33
➔ Inclure la dernière version de jquery.....	34
C) INSÉRER DU CODE CSS DANS LE HEAD.....	35
14-Les templates de page.....	36
A) CRÉATION D'UNE PAGE CONTACT.....	36
15-Les shortcodes.....	37
A) INSTALLATION DU PLUGIN CONTACT FORM 7.....	37
B) INSÉRER LE SHORTCODE PROPOSÉ PAR LE PLUGIN CONTACT FORM7.....	37
16-Un onepage avec wordpress.....	38
A) CRÉER UNE PAGE D'ACCUEIL STATIQUE ET UNE PAGE D'ACCUEIL DU BLOG.....	38
B) CRÉER DES PAGES ENFANTS.....	39
C) REQUÊTE PERSONNALISÉE : WP_QUERY.....	39
D) ET LE MENU DANS TOUT CELA ?.....	41
E) SURCHARGE DE LA FONCTION WORDPRESS GÉNÉRANT LE MENU : LES WALKERS.....	41
➔ Structure des template-parts des sections.....	43
➔ Insertion du walker dans le fichier de fonctions.....	43
➔ Modification de l'appel du menu afin de prendre en compte le walker.....	43
17-Les champs personnalisés.....	44
A) CRÉER UN CHAMP PERSONNALISÉ.....	44
B) AFFICHER LES CHAMPS PERSONNALISÉS.....	44
C) LIMITE FONCTIONNELLE.....	45
➔ Au niveau de l'administration.....	45
➔ Interface de saisie et Type de champ.....	45
➔ Code généré par la fonction the_meta().....	45
18-Le plugin Advanced Custom Field.....	46
A) CRÉER UN GROUPE DE CHAMP QUE NOUS NOMMERONS : RÉFÉRENCIEMENT.....	46
B) AJOUTER DES CHAMPS DANS LE GROUPE.....	55
➔ Les types de champs.....	55
C) LES NOUVEAUX CHAMPS DANS L'INTERFACE DE SAISIE.....	56
D) AFFICHAGE DANS LES TEMPLATES.....	56
➔ Désactiver l'ajout automatique du title dans le fichier de fonctions.....	56
➔ Exploiter les marqueurs conditionnels et afficher les champs ACF.....	56

LES CUSTOM POST TYPE.....57

1-Création d'un agenda.....	57
2-Les templates.....	58
A) PAGE D'ARCHIVE LISTANT LES ÉVÉNEMENTS.....	58
➔ Rajouter un lien vers l'agenda dans le menu.....	58
➔ Requête implicite du template archive-agenda.php.....	59
B) TEMPLATE SINGLE D'UN ÉVÉNEMENT.....	59
C) RAJOUTER DES CHAMPS AUX ÉVÉNEMENTS AVEC ACF.....	60
➔ Les champs supplémentaires.....	60
➔ Particularité pour les cartes Google.....	60
D) AFFICHER LES NOUVEAUX CHAMPS DANS LES TEMPLATES AGENDA.....	61
➔ Formater les dates.....	61
E) AFFICHER UNIQUEMENT LES ÉVÉNEMENTS À VENIR.....	63
F) CRÉER UNE PAGE POUR AFFICHER LES ÉVÉNEMENTS PASSÉS.....	64
➔ Faire un lien vers les pages.....	64
➔ Faire un lien vers une page d'archive.....	64
3-Les mu-plugins.....	65

A) DES PLUGINS SIMPLIFIÉS.....	65
B) FAIRE DE L'AGENDA UN MU-PLUGIN.....	65
C) FAIRE DU MENU ONEPAGE UN PLUGIN.....	66
D) FAIRE UN PLUGIN SIMPLIFIÉ POUR UN MENU DÉROULANT AU CLIC.....	66
➔ Enregistrer le script dropdown.js.....	66
➔ Hook pour rajouter un filtre dans le menu	67
➔ Insérer un lien personnalisé (dropdown) dans le menu WORDPRESS.....	67

PARTIR D'UN THÈME EXISTANT.....68

1-Les starter thèmes.....	68
2-Les thèmes enfants.....	68
A) STYLE.CSS.....	68
B) FUNCTIONS.PHP.....	68
C) LES TEMPLATES.....	69

ANNEXES.....70

1-Mockup page d'archives.....	70
2-Mockup single.....	71
3-Mockup page d'accueil onepage.....	72

PLAN DE TP ET OBJECTIF PÉDAGOGIQUE

1- Objectif de réalisation

Réaliser un thème de A à Z s'appuyant sur les fonctionnalités natives du CMS

Le stagiaire devra créer une interface aboutie en réalisant les templates suivants, afin d'appréhender au mieux les fonctionnalités

- ✓ template blog post (accueil du blog)
- ✓ template archive
- ✓ template post
- ✓ template page
- ✓ template page personnalisé

Mise en œuvre de fonctionnalités additionnelles

Le stagiaires devra appréhender le choix d'une extension WORDPRESS et son adéquation au besoin du projet en découvrant les possibilités d'extensions fonctionnelles natives

- ✓ Amélioration de l'interface d'administration des posts méta
- ✓ Création d'un nouveau post-type
- ✓ Création d'une nouvelle taxonomie
- ✓ Les requêtes personnalisées (Page d'accueil de type onepage)
- ✓ Recherche d'extensions (agenda, géolocalisation, galeries)

2-Plan de TP

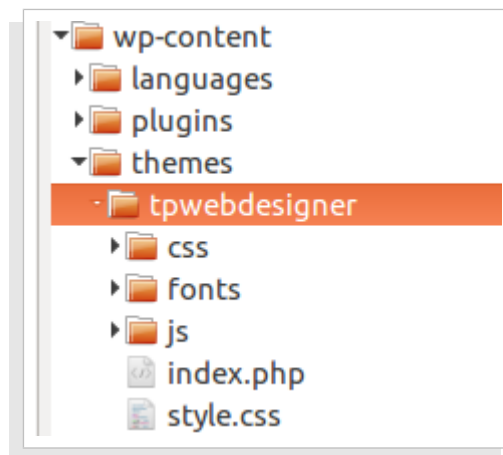
- Installation de Wordpress
- Outils et configuration
 - ✓ Réglages
 - ✓ Importation/Exportation
- Initialisation du thème
- Les marqueurs de modèles
- La boucle
- Le fichier de fonctions
 - ✓ Emplacement des menus
 - ✓ Barres de widgets
 - ✓ Image à la une des posts
 - ✓ Post-format des articles
 - ✓ Image d'en-tête paramétrable
 - ✓ Logo paramétrable
 - ✓ Fonctions personnalisées (hook)
 - ✓ Insertion de scripts
- Templates et hiérarchie des templates
 - ✓ Types de posts
 - ✓ Pages d'archives
 - ✓ Pages d'accueil (front-page et blog posts)
 - ✓ Gestions des médias et galerie de médias
- Découpage des templates, fonctions d'inclusions, template parts
- Les marqueurs conditionnels
- Les shortcodes
- Requêtes personnalisées
- Les champs personnalisés
- Custom post et custom taxonomy
- Thème enfant

RÉALISATION DU THÈME

1-Initialisation

A) Mise en place du dossier du thème

- 1) Création d'un répertoire **tpwebdesigner** dans le dossier **wp_content/themes**
- 2) Création d'un fichier **style.css** dans **wp_content/themes/ tpwebdesigner** (attention, le fichier doit être **à la racine du dossier du thème** et ne comporte **pas de s à la fin**)
- 3) Création d'un fichier index.php basé sur un document html de base (balises obligatoires + déclarations additionnelles)



B) Dans le fichier style.css

Rajouter les commentaires CSS suivants

```
/*
Theme Name: Formation TP WebDesigner
Author: Votre nom
Description: Création d'un thème WORDPRESS
Version: 0.0.1
*/
```

C) Activer le thème.

Dans l'interface d'administration, le thème doit être visible dans Apparence – Thème

En le survolant, cliquer sur le bouton **Activer**

2-Le fichier index.php

Insérer les premiers marqueurs de modèles

```
<!doctype html>
<html class="no-js" <?php language_attributes(); ?>
<head>
    <meta charset="<?php bloginfo( 'charset' ); ?>">
    <meta http-equiv="x-ua-compatible" content="ie=edge">
    <title><?php bloginfo( 'name' )?></title>
    <meta name="description" content="<?php bloginfo( 'description' )?>">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link rel="stylesheet" href="<?php bloginfo( 'template_directory' );?>/style.css">

</head>
<body <?php body_class(); ?> itemscope itemtype="http://schema.org/Website">

    <header class="site-header">
        <div class="container site-header-container">
            <!-- Logo et nav à venir -->
        </div>
    </header>

    <main>

        <header class="page-header container">
            <h1 class="page-title"><?php bloginfo( 'name' )?></h1>
            <p><?php bloginfo( 'description' )?></p>
        </header>

        <div class="site-main">

            <div class="site-main-inner container">

                <section>
                    <h2><?php echo esc_html( 'Les derniers articles' ) ?></h2>

                    <!-- Liste des articles à venir -->

                </section>
            </div>
        </div>

        <footer class="site-info">
            <div class="container">
                <a href="<?php echo esc_url( 'https://fr.wordpress.org/' ); ?>"><?php echo
esc_html( 'Propulsé par WordPress' ) ?></a>
            </div>
        </footer>

        <script src="<?php bloginfo( 'template_directory' );?>/js/main.js"></script>
    </body>
</html>
```

Voir en ligne :

- ✓ https://codex.wordpress.org/fr:Marqueurs_de_Modele
- ✓ https://codex.wordpress.org/Function_Reference/language_attributes
- ✓ <https://developer.wordpress.org/reference/functions/bloginfo/>
- ✓ https://codex.wordpress.org/Function_Reference/esc_url
- ✓ https://codex.wordpress.org/Function_Reference/esc_html

3-La boucle

```
<?php if ( have_posts() ) : ?>
    <section>
        <h2><?php echo esc_html( 'Les derniers articles' ) ?></h2>
        <!-- Liste des articles à venir -->
        <?php while ( have_posts() ) : the_post(); ?>
            <article itemscope itemtype="http://schema.org/Article">
                <!-- Contenu de chaque article à venir -->
            </article>
        <?php endwhile; ?>
    </section>
<?php endif; ?>
```

4-Les données de l'article

```
<!-- Contenu de chaque article à venir -->
<article id="post-<?php the_ID(); ?>"
    itemscope itemtype="http://schema.org/Article">
    <header>
        <h3 itemprop="name"><?php the_title() ?></h3>
        <p>Cet article est référencé dans : <span itemprop="about"><?php the_category('
        '); ?></span></p>
        <p class="post-meta">
            Posté le
            <time datetime="<?php echo get_the_date('Y-m-d'); ?>" itemprop="datePublished">
                <?php the_date(); ?>
            </time>
            par
            <a href="<?php the_author_url(); ?>">
                <span itemprop="author">
                    <?php the_author_firstname(); ?> <?php the_author_lastname(); ?>
                </span>
            </a>
        </p>
    </header>
    <div class="content" itemprop="description">
        <!-- Introduction : voir balise more -->
        <?php the_content('',true); ?>
    </div>
    <a class="btn" itemprop="url" href="<?php the_permalink(); ?>">
        <?php echo esc_html( 'Lire la suite' ) ?>
    </a>
</article>
```

A) Voir en ligne

Écrire un article : mode d'emploi

✓ <https://wpformation.com/article-wordpress/>

Les marqueurs :

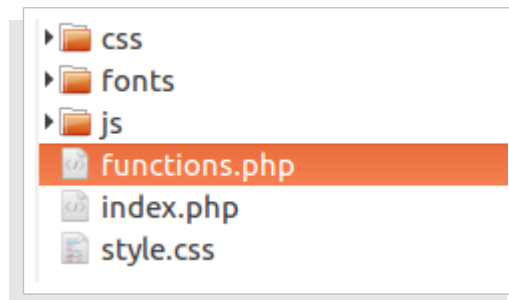
✓ https://codex.wordpress.org/fr:Marqueurs_de_Modele#Marqueurs_des_Articles
 ✓ https://codex.wordpress.org/Customizing_the_Read_More

Extrait personnalisé à la place de more :

✓ <https://codex.wordpress.org/fr:Extrait>
 ✓ <https://www.notuxedo.com/extraits-personnalisés-sur-wordpress/>

5-Activer des fonctionnalités : le fichier de fonctions

Créer un fichier **functions.php** à la racine du thème.



Le fichier functions.php

```
<?php
// Pas de fermeture de php
```

A) Permettre de personnaliser le thème depuis l'administration : add_theme_support

```
<?php
/* =====
   ACTIVATION DES FONCTIONNALITÉS
   https://developer.wordpress.org/reference/functions/add_theme_support/
   ===== */
/*
   Création d'un fonction personnalisée
   1/ Exécution de la fonction tpw_theme_support au chargement du thème :
       add_action( 'after_setup_theme', 'fonction à exécuter' );
   2/ Instruction de la fonction : tpw_theme_support()
*/

add_action( 'after_setup_theme', 'tpw_theme_support' );

function tpw_theme_support()
{
    // Plusieurs add_theme_support() à venir
}
```

➔ Image d'en-tête

```
function tpw_theme_support()
{
    // Plusieurs add_theme_support() à venir

    // Image d'en-tête
    add_theme_support( 'custom-header', array(
        'flex-width'     => true,
        'width'          => 1920,
        'flex-height'    => true,
        'height'         => 1285,
        'default-image' => get_template_directory_uri() . '/images/header.jpg',
    ) );
}
```

Une fois la fonction ajoutée :

- 1- Dans l'administration : voir theme -> personnaliser : rajouter une image d'en-tête
- 2- Dans le fichier index.php, rajouter le marqueur permettant d'afficher l'image d'en-tête :

```
<main>

    <header class="page-header container">
        <h1 class="page-title"><?php bloginfo('name')?></h1>
        <p><?php bloginfo('description')?></p>

        <?php the_custom_header_markup() ?>

    </header>
```

Observer le code HTML généré et l'emploi de l'attribut srcset

Responsive Images : srcset et sizes

Dans cet exemple, j'indique clairement au navigateur : "je dispose de deux images dont la première a une largeur de 320px et l'autre 640px, débrouille-toi avec ça pour piocher la meilleure".

```

```

Voir en ligne :

✓ <https://www.alsacreations.com/article/lire/1621-responsive-images-srcset.html>

➔ Personnaliser le logo du site

```
function tpw_theme_support()
{
    add_theme_support( 'custom-header' ...

    add_theme_support( 'custom-logo', array(
        'header-text' => array( 'site-title', 'site-description' ),
    ) );
}
```

Création d'une fonction personnalisée pour l'affichage si cette fonction existe

```
function tpw_theme_support()
{
    ----
}

function tpw_the_custom_logo() {
    if ( function_exists( 'the_custom_logo' ) )
    {
        the_custom_logo();
    }
}
```

Dans index.php

```
<header class="site-header">
  <div class="container site-header-container">
    <?php the_custom_logo(); ?>
  </div>
</header>
```

➔ *Balises html5 sur les contenus html générés par les fonctions*

```
function tpw_theme_support()
{
    add_theme_support( 'custom-header', array(
        ---
    ) );

    add_theme_support( 'custom-logo', array(
        ---
    ) );

    add_theme_support( 'html5', array( 'comment-list', 'comment-form', 'search-form',
    'gallery', 'caption' ) );
}
```

➔ *Image à la une sur les posts*

Pouvoir choisir une image mise en avant sur les articles ou les pages (les posts peuvent être associés à plusieurs médias)

```
function tpw_theme_support()
{
    add_theme_support( 'custom-header' ...
    add_theme_support( 'custom-logo' ...
    add_theme_support( 'html5' ...

    add_theme_support( 'post-thumbnails' );
}
```

1. Rajouter une image à la une sur un article dans l'administration
2. Voir le répertoire wp_content/upload : l'image existe en plusieurs formats
3. Puis, dans index.php, afficher la version miniature

```
<?php while ( have_posts() ) : the_post(); ?>
  <article id="post-<?php the_ID(); ?>" itemscope itemtype="http://schema.org/Article">
    <header>
      ...
    </header>

    <?php if ( has_post_thumbnail() ) : ?>
      <?php the_post_thumbnail( 'thumbnail' ); ?>
    <?php endif; ?>

    <div class="content" itemprop="description">
      ...
    </div>
  </article>
<?php endwhile; ?>
```

✓ Voir en ligne : https://codex.wordpress.org/fr:Marqueurs_de_Modele/the_post_thumbnail

B) Permettre de personnaliser des menus depuis l'administration : register_nav_menus()

A la suite du fichier functions.php, déclarer deux **emplacements de menu**

```
/* =====
EMPLACEMENT DE MENU
https://codex.wordpress.org/Function_Reference/register_nav_menus
===== */

add_action( 'init', 'tpw_register_menus' );

function tpw_register_menus() {

    register_nav_menus( array (

        'header' => __( 'Menu principal' ),
        'footer' => __( 'Pied de page' ),

    ) );

}
```

Nota : une autre fonction register_nav_menu() fonctionne à l'identique, mais elle ne permet de prévoir qu'un seul emplacement de menu

1. Dans l'interface d'administration : créer 2 menus et les placer respectivement dans chaque emplacement

1 - Donner un nom au menu

Nom du menu : Menu du header

Structure du menu

Glissez chaque élément pour les placer dans l'ordre que vous préférez. Cliquez sur la flèche à droite de l'élément pour afficher d'autres options de configuration.

Non classé Catégorie

Règlages du menu

Ajoutez automatiquement des pages

☐ Ajouter automatiquement les pages de premier niveau à ce menu

Afficher l'emplacement

☒ Menu principal

☐ Pied de page

Synchroniser le menu

2 - Faire glisser les pages désirées dans le menu

3 - Cocher l'emplacement prévu dans le thème

2. Dans index.php, insérer le menu avec la fonction wp_nav_menu()

```
<header class="site-header">
    <div class="container site-header-container">

        <?php the_custom_logo(); ?>

        <?php wp_nav_menu( array(
            'theme_location' => 'header' ,
            'container' => 'nav' ,
            'container_id' => 'navigation' ,
            'container_class' => 'menu-header' ,
        ) ); ?>

    </div>
</header>
```

- ✓ https://codex.wordpress.org/Navigation_Menus
- ✓ <https://wpmarmite.com/menu-wordpress/>

C) Créer une fonction pour rajouter un lien vers l'accueil automatiquement

Cette fonction va être interceptée par la fonction qui génère le menu dans l'emplacement *header* et rajouter un lien menant vers la page d'accueil.

Nota : au passage, noter le *code html généré*

```
function tpw_register_menus() {
    register_nav_menus( array (

        'header' => __( 'Menu principal' ),
        'footer' => __( 'Pied de page' ),

    ) );
}

add_filter('wp_nav_menu_items', 'tpw_add_index_link', 10, 2);

function tpw_add_index_link($items, $args) {

    if( $args->theme_location == 'header' )
        $homeLink .= '<li id="menu-item-home" class="menu-item menu-item-home"><a href="' .
        home_url() . '">Accueil</a></li>';

    return $homeLink . $items;
}
```

D) Prévoir l'ajout de widgets dans une sidebar : register_sidebar()

A la suite du fichier de fonctions.php, rendre le thème « widgétisable »

La fonction prévoit un emplacement pour une barre de widgets (même principe que pour les menus)

```
/* =====
    BARRE DE WIDGETS
    https://codex.wordpress.org/Function_Reference/register_sidebar
===== */

add_action( 'widgets_init', 'tpw_widgets_init' );

function tpw_widgets_init() {

    register_sidebar( array(
        'name' => __( 'Barre latérale', 'tpwebdesigner' ),
        'id' => 'sidebar',
        'description' => __( 'Menu de widgets', 'tpwebdesigner' ),
        'before_widget' => '<div id="%1$s" class="widget %2$s">',
        'after_widget' => '</div>',
        'before_title' => '<h3 class="widget-title">',
        'after_title' => '</h3>',

    ) );
}
```

Dans le fichier index.php, après la boucle

```
<?php if ( is_active_sidebar( 'sidebar' ) ) : ?>
    <aside class="sidebar">
        <?php dynamic_sidebar( 'sidebar' ); ?>
    </aside>
<?php endif; ?>
```

Rajouter des widgets dans la sidebar depuis l'administration

6-Permettre à des plugins des déclarations supplémentaires dans le code html

Les plugins (extensions) chargent généralement des feuilles de styles et des javascripts

Dans index.php

La fonction **wp_head()** permet l'insertion de déclarations supplémentaires dans le **head**

```
<head>
  <meta charset="<?php bloginfo( 'charset' ); ?>">
  ...
  <link rel="stylesheet" href="<?php bloginfo( 'template_directory' );?>/style.css">

  <?php wp_head(); ?>

</head>
```

La fonction **wp_footer()** permet l'insertion de javascript ou de code html à la fin du document

```
<?php wp_footer(); ?>

<script src="<?php bloginfo( 'template_directory' );?>/js/main.js"></script>
</body>
</html>
```

➔ Observer l'apparition d'une barre de navigation en haut de la page :

Le code de ce menu est généré à la fin du document, la css est dans le head.

Il disparaît à la déconnexion : il n'est pas visible par le public non connecté.

Si on souhaite fixer la bannière, il sera nécessaire de savoir si on est logé ou pas.

Observer les classes rajoutées avec la fonction `<?php body_class(); ?>` dans la balise body

Une **class=logged-in** est générée si l'utilisateur est connecté

En css, il est donc possible de modifier le style des éléments concernés

```
// Marge de haut ajouté à body si connecté
.logged-in {
  margin-top : 32px;
}

.site-header {
  position : fixed;
  top : 0; left : 0; right : 0;
}

// Position de la bannière modifiée si connecté
.logged-in .site-header {
  top : 32px;
}
```

7-Découper le template

A) Créer les fichiers suivants à la racine du thème :

- ✓ header.php
- ✓ footer.php
- ✓ sidebar.php

B) Créer un dossier templates-part à la racine du thème

C) Créer les fichiers suivants à l'intérieur du dossier templates-part

- ✓ content.php

D) Découper le template

- ✓ Dans header.php : tout depuis le début jusqu'à l'ouverture de la balise main exclue

```
<!doctype html>
<html class="no-js" <?php language_attributes(); ?>
  <head>
    .....
  </head>
  <body <?php body_class(); ?> itemscope itemtype="http://schema.org/Website">

    <header class="site-header">
      <div class="container site-header-container">
        <?php the_custom_logo(); ?>
        <?php wp_nav_menu( array(
          'theme_location' => 'header' ,
          'container' => 'nav' ,
          'container_id' => 'navigation' ,
          'container_class' => 'menu-header' ,
        ) ); ?>
      </div>
    </header>
```

- ✓ Dans footer.php : tout depuis la fermeture de la balise main exclue jusqu'à la fin

```
<footer class="site-info">
  <div class="container">
    <a href="<?php echo esc_url('https://fr.wordpress.org/'); ?>"><?php echo
esc_html( 'Propulsé par WordPress' ) ?></a>
  </div>
</footer>

<?php wp_footer(); ?>

<script src="<?php bloginfo('template_directory'); ?>/js/main.js"></script>
</body>
</html>
```

- ✓ Dans sidebar.php : la barre de widget

```
<?php if ( is_active_sidebar( 'sidebar' ) ) : ?>
<aside class="sidebar">
  <?php dynamic_sidebar( 'sidebar' ); ?>
</aside>
<?php endif; ?>
```

✓ Dans content.php : les données de l'article

```
<article id="post-<?php the_ID(); ?>" itemscope itemtype="http://schema.org/Article">
  <header>
    <h3 itemprop="name"><?php the_title() ?></h3>
    <p>Cet article est référencé dans : <span itemprop="about"><?php the_category(', '); ?></span></p>
    <p class="post-meta">
      Posté le <time datetime="<?php echo get_the_date('Y-m-d'); ?>"
      itemprop="datePublished"><?php the_date(); ?></time>
      par <a href="<?php the_author_url(); ?>"><span itemprop="author"><?php
      the_author_firstname(); ?> <?php the_author_lastname(); ?></span></a>
    </p>
  </header>
  <?php if ( has_post_thumbnail() ) : ?>
    <?php the_post_thumbnail( 'thumbnail' ); ?>
  <?php endif; ?>
  <div class="content" itemprop="description">
    <?php the_content('',true); ?>
  </div>
  <a class="btn" itemprop="url" href="<?php the_permalink(); ?>"><?php echo esc_html( 'Lire
  la suite' ) ?></a>
</article>
```

E) Inclure les parties extraites dans le template

➔ *Header, footer, sidebar*

Les fonctions réalisent un `include()` des fichiers respectifs, lesquels doivent se trouver dans le même répertoire :

- ✓ `get_header()` : inclure un fichier `header.php`
- ✓ `get_footer()` : inclure un fichier `footer.php`
- ✓ `get_sidebar()` : inclure un fichier `sidebar.php`

En l'absence des fichiers correspondants, ces fonctions recherchent les fichiers dans le thème du moteur par défaut : `wp-includes/theme-compat/`

➔ *template-parts*

La fonction **`get_template_parts()`** permet, en revanche, de spécifier un dossier du thème.

Le dossier **`template-parts`** va, en fait, contenir différentes versions de présentation des posts. (voir

A l'arrivée, le template index.php :

```
<?php get_header(); ?>

<main>

<header class="page-header container">
  <h1 class="page-title"><?php bloginfo('name')?></h1>
  <p><?php bloginfo('description')?></p>
  <?php the_custom_header_markup() ?>
</header>

<div class="site-main">

  <div class="site-main-inner container">

    <?php if ( have_posts() ) : ?>
      <section>
        <h2><?php echo esc_html( 'Les derniers articles' ) ?></h2>

        <?php while ( have_posts() ) : the_post(); ?>

          <?php get_template_part( 'template-parts/content' ); ?>

        <?php endwhile; ?>
      </section>
    <?php endif; ?>

    <?php get_sidebar(); ?>

  </div>
</div>
</main>

<?php get_footer(); ?>
```

F) Les post formats

Les Posts Formats permettent de subdiviser les articles en plusieurs types afin de leur appliquer un style différent en choisissant parmi l'un des types suivants :

- ✓ aside : article rapide (sans titre)
- ✓ chat : une discussion
- ✓ gallery : une galerie d'images affichant les images d'un post
- ✓ link : Une présentation d'un ou plusieurs liens externes
- ✓ image : une présentation d'une simple image
- ✓ quote : une citation
- ✓ status : un statut, comme Twitter ou Facebook par exemple
- ✓ video : une vidéo
- ✓ audio : un podcast, une musique ou autre sons

L'intérêt des post formats est de pouvoir personnaliser la mise en page des articles en fonction de ce type.

Dans la même boucle, les articles pourront se présenter différemment.

➔ Activer les posts formats pour le thème

Pour activer tous les posts-formats :

```
add_theme_support( 'post-formats', array( 'aside', 'link', 'gallery', 'status', 'quote',
'image' ) );
```

Mais il est possible de n'activer que ceux utiles

```
function tpw_theme_support()
{
    // Image d'en-tête
    add_theme_support( ... );

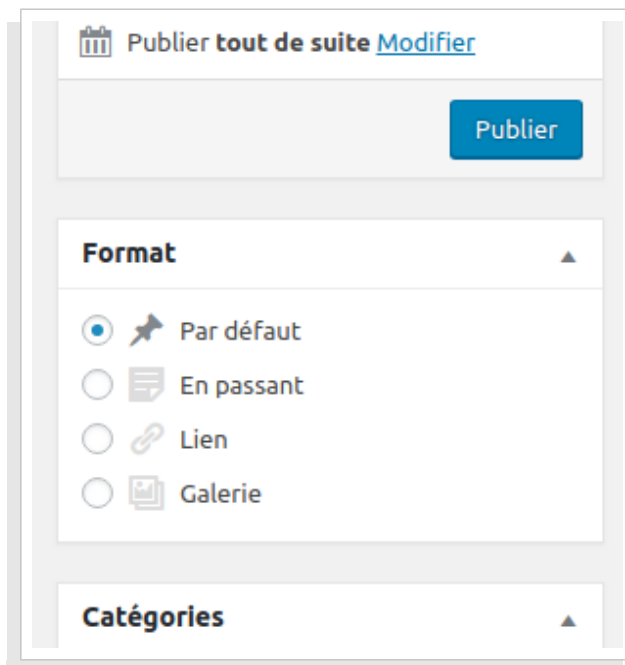
    // Logo
    add_theme_support( ... );

    // html5 sur html généré
    add_theme_support( ... );

    // Image à la une sur les posts
    add_theme_support( 'post-thumbnails' );

    // 3 post-formats
    add_theme_support( 'post-formats', array( 'aside', 'link', 'gallery' ) );
}
```

➔ paramétrer un post format sur un article



The screenshot shows the WordPress post editor interface. At the top, there is a 'Publier tout de suite' button and a 'Modifier' link. Below this is a 'Publier' button. The 'Format' dropdown menu is open, showing four options: 'Par défaut' (selected), 'En passant', 'Lien', and 'Galerie'. Below the 'Format' menu is the 'Catégories' dropdown menu.

➔ *Un template-part par post-format*

Dans le fichier index.php, à l'intérieur de la boucle, il est possible de récupérer la valeur du post-format de l'article avec la fonction : **get_post_format()**

get_post_format() renvoie la valeur :

- ✓ aside : si le post-format « *En passant* » est sélectionné
- ✓ gallery : si le post-format « *Galerie* » est sélectionné
- ✓ link : si le post-format « *Lien* » est sélectionné
- ✓ etc.

get_template_part() admet deux paramètres :

1. le chemin du fichier à inclure (obligatoire)
2. un suffixe éventuel (facultatif)

get_template_part('template-parts/content', 'aside')

va inclure le contenu du fichier

./template-parts/content-aside.php

Dans la boucle, pour chaque article, il suffit de charger le template-part correspondant :

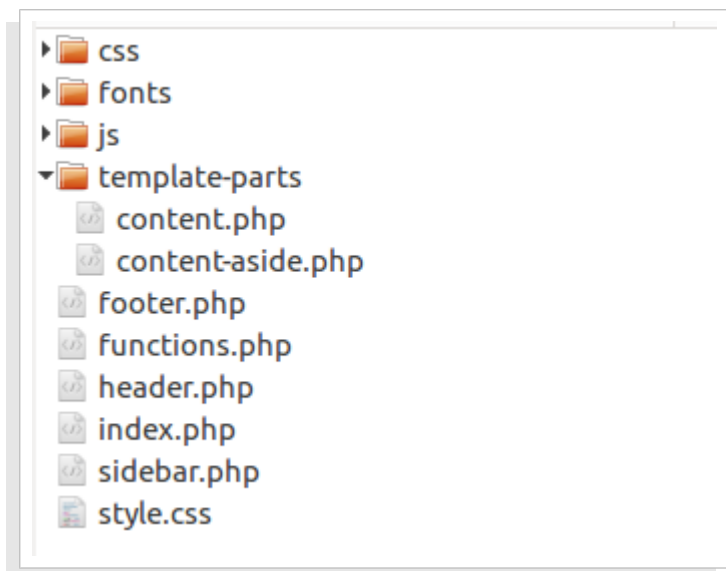
```
<?php while ( have_posts() ) : the_post(); ?>

    <?php get_template_part( 'template-parts/content' , get_post_format() ); ?>

<?php endwhile; ?>
```

Il suffit de réaliser autant de fichiers template-parts que de post formats activés :

- ✓ content-aside.php
- ✓ content-link.php
- ✓ etc.



Voir en ligne :

- ✓ https://codex.wordpress.org/Function_Reference/get_post_format
- ✓ <http://www.geekpress.fr/post-formats-wordpress/>

8- Les templates à réaliser

A) index.php

Il s'agit du template par défaut du thème. Si le fichier **index.php** est le seul template du thème, toutes les pages du site l'utiliseront.

B) archive.php

Il s'agit d'une liste d'articles répondant à certaines conditions. Les pages dites « statiques » (appelées simplement « pages » dans le back-office) ne sont pas listées dans les archives.

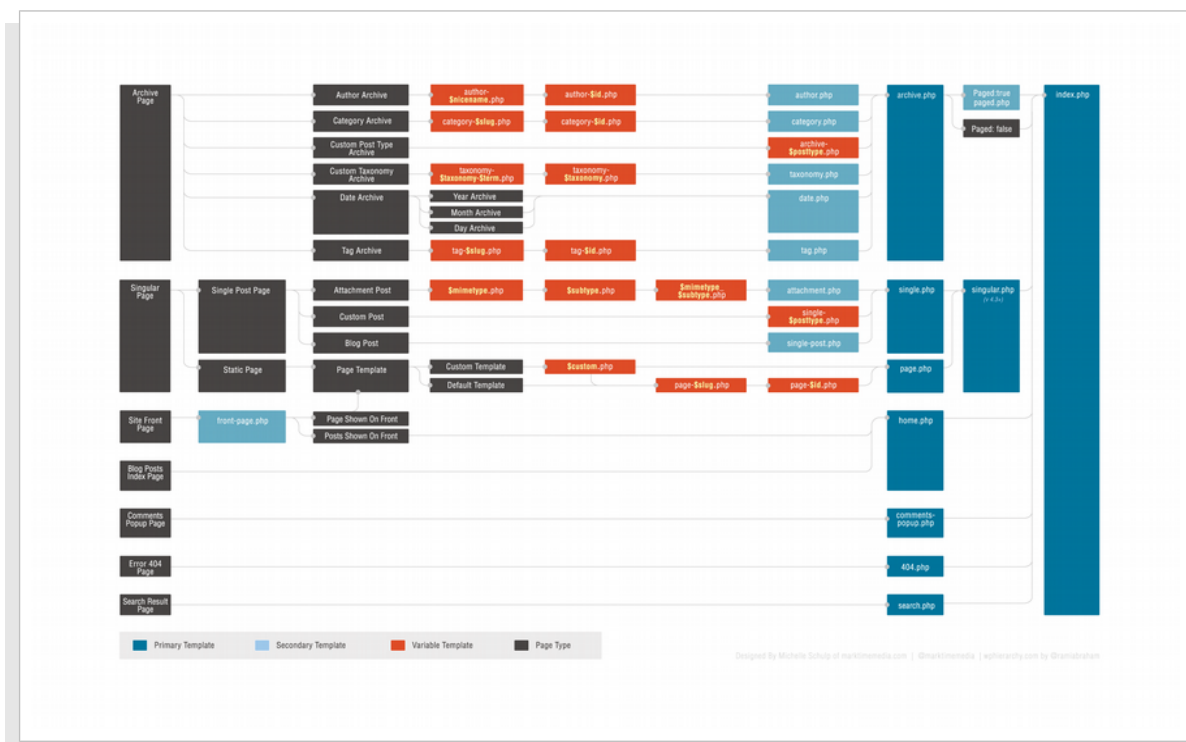
C) single.php

C'est le template par défaut de tous les posts qui ne sont pas des pages : (articles, médias et custom posts)

D) page.php

Template standard pour afficher les pages statiques

9- La hiérarchie des templates



10-Les commentaires

Insérer les commentaires dans les templates d'articles si les commentaires sont ouverts ou si il existe des commentaires postés

```
<?php if ( comments_open() || get_comments_number() ) : ?>
<?php comments_template(); ?>
<?php endif; ?>
```

A) Le template comments.php

La fonction `comments_template()` utilise un template nommé **comments.php**

A défaut, le template utilisé est celui situé dans le répertoire **wp_includes/theme_compat**

B) Gestion des commentaires

<http://deligraph.com/tuto-wordpress/gerer-les-commentaires-wordpress/>

C) Les rétroliens

<https://blogpascher.com/blog-2/comment-traiter-les-retroliens-trackbacks-et-pingbacks-dans-wordpress-pret>

11-Le moteur de recherche

Le moteur de recherche correspond à un template nommé **searchform.php**

```
<form method="get" class="search-form" id="searchform" action="<?php bloginfo('home'); ?>"/>
  <label for="recherche"><span class="visuallyhidden">Recherche</span></label>
  <input type="search" id="recherche" value="<?php echo wp_specialchars($s, 1); ?>"
  name="s" placeholder="Rechercher" >
  <input type="submit" value="Ok">
</form>
```

Pour l'afficher dans un template

```
<?php get_search_form() ?>
```

Les résultats de la recherche ont un template spécifique : **search.php**

La boucle affiche la liste des posts correspondant aux résultats de la recherche.

A défaut, **index.php** est utilisé.

12-Les médias sous WORDPRESS

<https://wpformation.com/ajouter-medias-wordpress/>

WordPress redimensionne automatiquement les médias visuels téléchargés.

Une image téléchargée est déclinée en 5 formats par défaut :

- | | |
|---------------------|-----------------------|
| 1. taille originale | → full |
| 2. miniature | → thumbnail |
| 3. moyenne | → medium |
| 4. moyenne large | → medium-large |
| 5. grande taille | → large |

Les formats de redimensionnements sont paramétrables via **Réglages/Médias**

A) Installation du plugin Simple Image Sizes

Ce plugin utilitaire permet de visualiser les formats existants, d'en créer d'autres mais surtout de traiter les images déjà téléchargées si un nouveau format a été créé entre temps.

<https://fr.wordpress.org/plugins/simple-image-sizes/>

B) Créer de nouveaux formats

Il est possible de rajouter de nouveaux formats à l'aide de la fonction `add_image_size()` qui admet 4 paramètres :

1. le nom du format
2. la largeur
3. la hauteur
4. option de recadrage définie de la manière suivante
 - ✓ `false` → pas de recadrage (par défaut)
 - ✓ `true` → recadrage au centre ou un tableau d'arguments pour définir les options de recadrage

Rajout d'un nouveau format dans le fichier de fonctions, dans la fonction existante `tpw_theme_support()`

```
function tpw_theme_support()
{
    ...

    /*
     Créer un format d'image personnalisée, recadrée au centre horizontal et en partant du
     haut
    */
    add_image_size( 'paysage', 1024, 300, array( 'center', 'top' ) );

    ...
}
```

C) Affichage d'un visuel en fonction d'un format

Affichage d'une image à la une dans un template utilisant ce nouveau format

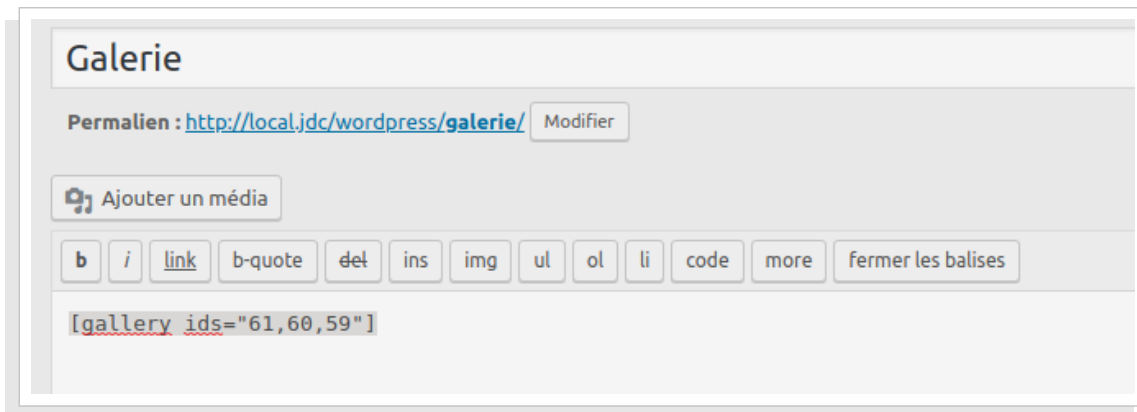
```
<?php the_post_thumbnail( 'paysage' ); ?>
```

D) Insérer une galerie dans un article

<https://www.gregoirenoyelle.com/wordpress-integrer-une-galerie-photo/>

L'insertion de la galerie correspond à un code appelé **shortcode** listant les ids des médias dans la base.

La taille par défaut des photos est la miniature





Il est possible de modifier la taille via l'insertion de la galerie ou directement dans le shortcode

Pour aller plus loin : https://codex.wordpress.org/Gallery_Shortcode

E) Installer un plugin lightbox

<https://archetyped.com/tools/simple-lightbox/>

Il existe de nombreuses extensions wordpress permettant d'améliorer la galerie native, mais aussi des extensions plus élaborées qui installent un nouveau type de posts.

Le choix dépendra toujours des besoins fonctionnels ; il est préférable d'éviter de choisir une extension avec comme seul critère le choix graphique dans la mesure où nous sommes en mesure d'intervenir sur ce plan.

13-Insertion de css et de scripts via le fichier de fonctions

Les fonctions **wp_head()** et **wp_footer()** insérées dans les templates permettent l'insertion de styles ou de scripts à partir d'une extension ou du fichier de fonctions d'un thème.

Penser à supprimer les déclarations dans les templates !

A) Insérer des feuilles de styles

2 fonctions pour les styles :

1. **wp_register_style()** : déclarer la feuille
2. **wp_enqueue_style()** : l'envoyer à la fonction d'insertion

B) Insérer des scripts

2 fonctions pour les scripts :

3. **wp_register_script()** : déclarer le script
4. **wp_enqueue_script()** : l'envoyer à la fonction d'insertion

```
/*
  Insérer des scripts (via wp_footer ou wp_head)
*/

add_action( 'wp_enqueue_scripts', 'tpw_js' );

function tpw_js(){

    // Insérer des feuilles de styles

    wp_register_style( 'fonts', get_template_directory_uri() . '/css/fonts.css' );
    wp_enqueue_style( 'fonts' );

    // Insérer des scripts

    wp_register_script(
        'main', // Nom du script
        get_template_directory_uri() . '/js/main.js', // URL du script
        array(), // Dépendance avec d'autres script ( chaque valeur du tableau contient le nom
        donné à un autre script )
        '1.0', // Numéro de version du script
        true // Chargement via wp_footer (par défaut : chargement dans le head avec wp_head)
    );

    wp_enqueue_script( 'main' );

}
```

➔ *Inclure la dernière version de jquery*

jQuery est déjà chargé dans Wordpress, mais il ne s'agit pas de la dernière version.

La dernière version ne sera chargée que sur la partie publique (la version jQuery de Wordpress doit être conservée dans l'administration afin d'éviter d'éventuels conflits)

Récupérer la dernière version de jQuery dans le dossier js/vendor/jquery-3.2.1.min.js

Si le script main.js repose sur jquery, on l'indiquera dans son tableau de dépendance.

```
add_action( 'wp_enqueue_scripts', 'tpw_js' );
function tpw_js()
{
    wp_register_style( 'fontello', get_template_directory_uri() . '/css/fontello.css' );
    wp_enqueue_style( 'fontello' );

    wp_register_script(
        'main', // Nom du script
        get_template_directory_uri() . '/js/main.js', // URL du script
        array( 'jquery' ), // Dépendance avec d'autres script
        '2.0', // Numéro de version du script
        true // Chargement via wp_footer (par défaut : chargement dans le head avec wp_head)
    );

    /*
     * Charger la dernière version de jQuery
     * Si on n'est pas dans l'admin
     */

    if( !is_admin() )
    {
        // Supprimer la version WORDPRESS
        wp_deregister_script( 'jquery' );

        // Déclarer la version 3.2.1 dans wp_footer
        wp_register_script(
            'jquery',
            get_template_directory_uri() . '/js/vendor/jquery-3.2.1.min.js',
            array(),
            '3.2.1',
            true);

        wp_enqueue_script( 'jquery' );
    }

    wp_enqueue_script( 'smooth' );
    wp_enqueue_script( 'main' );
}
```

<https://www.alsacreations.com/astuce/lire/942-derniere-version-jquery-wordpress.html>

C) Insérer du code CSS dans le head

L'exemple suivant permet de déclarer le custom header en background plutôt que par une insertion de l'image dans le code html

Il faut d'abord prévoir des images par défaut à stocker dans le répertoire CSS du thème.

On créera 3 versions de l'image pour une photo destinée à occuper la largeur du viewport :

1. **header-768.jpg** → largeur : 768px
2. **header-1024.jpg** → largeur : 1024px
3. **header.jpg** → largeur : 1920px

Ces trois formats correspondent aux tailles d'images par défaut de Wordpress suivantes :

1. medium
2. large
3. full

La fonction `tpw_customize_css()` est destinée à insérer dans le head des templates une feuille de styles afin de charger une image de fond dans la balise body en fonction de la taille du viewport.

Si une image d'en-tête personnalisée a été choisie, cette dernière sera récupérée, sinon on chargera les images par défaut stockées dans le répertoire du thème.

```
add_action( 'wp_head', 'tpw_customize_css' );
function tpw_customize_css()
{
    $custom_header = get_custom_header();

    if ( ! empty( $custom_header->attachment_id ) )
    {
        $medium = wp_get_attachment_image_url( $custom_header->attachment_id,
'medium_large' );
        $large = wp_get_attachment_image_url( $custom_header->attachment_id, 'large' );
        $full = wp_get_attachment_image_url( $custom_header->attachment_id, 'full' );
    }
    else
    {
        $medium = get_template_directory_uri() . '/images/header-768.jpg';
        $large = get_template_directory_uri() . '/images/header-1024.jpg';
        $full = get_template_directory_uri() . '/images/header.jpg';
    }

    //On ferme php ?>
    <style>
        @media only screen and (max-width: 768px) {
            body {
                background-image : url(<?php echo $medium; ?>);
            }
        }
        @media only screen and (min-width: 769px) and (max-width: 1024px) {
            body {
                background-image : url(<?php echo $large; ?>);
            }
        }
        @media only screen and (min-width: 1025px) {
            body {
                background-image : url(<?php echo $full; ?>);
            }
        }
    </style>
    <?php // On ré-ouvre PHP
}
```

14-Les templates de page

Les templates de page sont des templates personnalisés qu'il est possible de nommer de façon libre à condition de ne pas utiliser un nom réservé.

<http://wordpress.bbxdesign.com/templates-de-page>

A) Création d'une page contact

1. Créer un template contact.php à la racine du thème
2. Rajouter le commentaire suivant : **Template Name: contact**
3. Créer une nouvelle page depuis l'interface d'administration et lui affecter ce modèle qui apparaît dans l'encart **Attribut de page** dans la liste de sélection **Modèle**

```
<?php
/*
  Template Name: contact
  http://wordpress.bbxdesign.com/templates-de-page
*/

get_header(); ?>

<main>

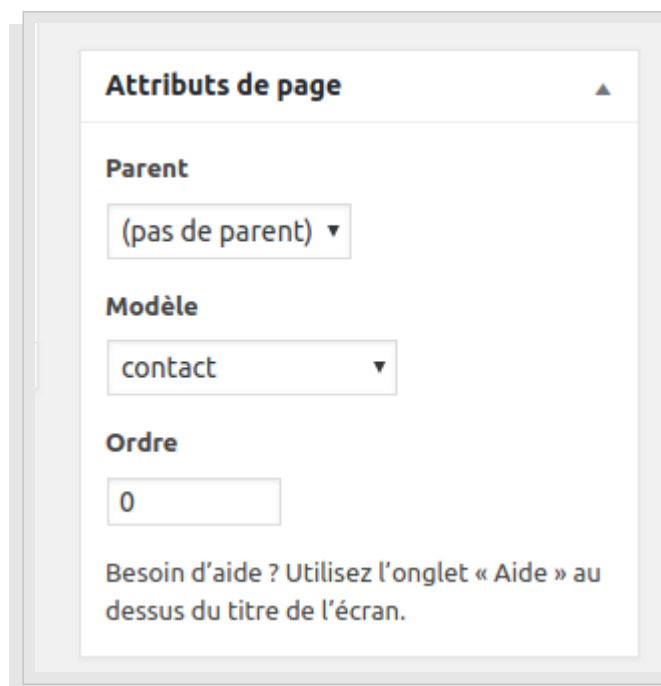
  <header class="page-header">
    <div class="container page-header-container">
      <h1><?php wp_title('') ?></h1>
    </div>
  </header>

  <div class="site-main">

    <?php while ( have_posts() ) : the_post(); ?>
      <?php get_template_part( 'template-parts/content' , 'page' ); ?>
    <?php endwhile; ?>

  </div>
</main>

<?php get_footer(); ?>
```



Attributs de page

Parent

(pas de parent) ▼

Modèle

contact ▼

Ordre

0

Besoin d'aide ? Utilisez l'onglet « Aide » au dessus du titre de l'écran.

15-Les shortcodes

Les shortcodes sont des petits morceaux de code entre crochets qu'il est possible d'insérer dans les articles ou les pages.

Ils fonctionnent comme un raccourci, en faisant la liaison entre l'éditeur WordPress et la fonction PHP qui se trouve derrière.

Ils offrent généralement personnalisables et ne nécessitent pas une grande connaissance en développement.

De nombreux plugins exploitent cette fonctionnalité .

A) Installation du plugin Contact Form 7

<https://fr.wordpress.org/plugins/contact-form-7/>

B) Insérer le shortcode proposé par le plugin contact form7

A l'installation du plugin contact form7, un formulaire de contact est déjà créé :



Il suffit d'insérer le shortcode dans le formulaire de saisie de la page contact :



16-Un onepage avec wordpress

Ce TP va permettre de mettre en application les fonctionnalités suivantes :

1. Créer une page d'accueil statique qui sera construite comme une onepage
2. Créer une page de blog pour afficher la liste des articles
3. Exploiter une requête personnalisée
4. Modifier le menu (afin de modifier les urls pour pointer sur des ancres) en surchargeant la fonction Wordpress qui le génère avec un customWalker fourni.

A) Créer une page d'accueil statique et une page d'accueil du blog

1. Créer un template de page que nous nommerons **onepage.php**
2. Créer une nouvelle page nommée **Accueil** via l'administration et lui affecter ce modèle
3. Créer une nouvelle page nommée **Blog** : cette page sera un peu spéciale et devra s'appuyer sur un template **home.php** à l'image du premier modèle **index.php**
4. Dans Réglages / Lecture :

home.php

```
<?php get_header(); ?>
<main>
  <header class="page-header">
    <div class="container page-header-container">
      <h1><?php wp_title('') ?></h1>
    </div>
  </header>
  <div class="site-main">
    <div class="site-main-inner container">
      <?php if ( have_posts() ) : ?>
        <section>
          <h2><?php echo esc_html( 'Les derniers articles' ) ?></h2>

          <?php while ( have_posts() ) : the_post(); ?>

            <?php get_template_part( 'template-parts/content' , get_post_format() ); ?>

          <?php endwhile; ?>
        </section>
      <?php endif; ?>
      <?php get_sidebar(); ?>
    </div>
  </div>
</main>
<?php get_footer(); ?>
```

onepage.php

```

<?php
/*
  Template Name: onepage
*/

get_header(); ?>

<main>

  <header class="page-header">
    <div class="container page-header-container">
      <h1 class="page-title"><?php bloginfo('name')?></h1>
      <p class="description"><?php bloginfo('description')?></p>
    </div>
    <?php tpw_custom_header() ?>
  </header>

  <div class="site-main">

    <?php while ( have_posts() ) : the_post(); ?>
      <?php get_template_part( 'template-parts/content' , 'page' ); ?>
    <?php endwhile; ?>
  </div>

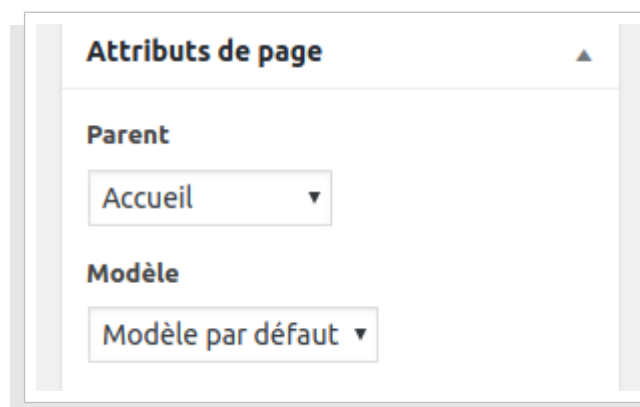
</main>

<?php get_footer(); ?>

```

B) Créer des pages enfants

Créer autant de page que de sections souhaitées depuis l'administration en les déclarant comme page enfant de la page Accueil



C) Requête personnalisée : wp_query

Il est possible d'influencer la requête principale ou de créer une requête en supplément.

wp_query() offre de très nombreuses possibilités d'extraction de données, à partir d'un tableau d'arguments ; à chaque besoin, il s'agira d'étudier les possibilités.

- ✓ <http://www.geekpress.fr/wp-query-creez-des-requetes-personnalisees-dans-vos-themes-wordpress/>
- ✓ https://codex.wordpress.org/Class_Reference/WP_Query
- ✓ https://codex.wordpress.org/Class_Reference/WP_Query#Post_26_Page_Parameters

Dans le template **onpage.php** qui sert de modèle à la page **Accueil**, nous allons exécuter une requête personnalisée afin de récupérer les pages enfants.

Le tableau d'arguments est le suivant :

1. Récupération de pages
2. Dont la page parente correspond à la page courante (\$post->ID renvoie l'id de la page en cours)
3. Classée par ordre ascendant
4. Dans l'ordre défini par les attributs de pages

```
<?php
/*
  Template Name: onpage

  Requête personnalisée
*/

$args = array(
  'post_type'      => 'page',
  'post_parent'    => $post->ID,
  'order'          => 'ASC',
  'orderby'        => 'menu_order'
);

// Exécution de la requête : $query contient les pages enfants
$query = new WP_Query( $args );

get_header(); ?>

...
```

A partir de là, nous pourrions faire deux boucles :

1. la boucle standard qui correspond aux données de la page en cours
2. une boucle sur \$query

```
...
<div class="site-main">

  <?php while ( have_posts() ) : the_post(); ?>
  <?php get_template_part( 'template-parts/content' , 'page' ); ?>
  <?php
  endwhile;
  wp_reset_postdata(); // Remise à 0 du curseur de requête principale (closeCursor())
  ?>

  <?php if ( $query->have_posts() ) : ?>
    <?php while ( $query->have_posts() ) : $query->the_post(); ?>

      <?php get_template_part( 'template-parts/content' , 'page' ); ?>

    <?php endwhile; ?>
  <?php endif; ?>

</div>

...
```


D) Et le menu dans tout cela ?

Nous allons réaliser un menu pointant vers des ancres à l'intérieur de la page, mais permettant aussi de rajouter d'autres liens, comme celui pointant sur la page blog.

Depuis l'administration, créer le menu normalement, en insérant les pages externes et les pages enfants exploitées en sections :

Nom du menu

Structure du menu

Glissez chaque élément pour les placer dans l'ordre que vous préférez. Cliquez sur la flèche à droite de l'élément pour aff

Accueil	Page ▼
Section 1	Page ▼
Section 2	Page ▼
Contact	Page ▼
Blog	Page ▼

Réglages du menu

Ajoutez automatiquement des pages ☐ Ajouter automatiquement les pages de premier niveau à ce menu

Afficher l'emplacement ☒ Menu principal ☐ Pied de page

E) Surcharge de la fonction Wordpress générant le menu : les walkers

Le rôle d'un walker est de redéfinir le comportement des fonctions générant des listes. Ils sont possibles avec les fonctions suivantes :

1. **wp_nav_menu()** : fonction générant les menus
2. **wp_list_pages()** : fonction générant des listes de pages
3. **wp_list_categories()** : fonction générant des listes de catégories
4. **wp_list_comments()** : fonction générant la liste des commentaires

<https://wabeo.fr/construire-walker-wordpress/>

<https://wpastuces.com/239-wordpress-menu-le-guide-complet/>

De nombreux walkers sont proposés sur le web, mais chacun va répondre à un besoin spécifique et doit souvent être adapté.

Le walker de notre menu est fourni via le fichier **customWalker.php** que nous stockerons dans un dossier nommé **inc** à la racine du thème.

customWalker.php (fourni)

```

<?php

class Single_Page_Walker extends Walker_Nav_Menu{
    function start_el(&$output, $item, $depth = 0, $args = array(), $id = 0) {
        global $wp_query;
        $indent = ( $depth ) ? str_repeat( "\t", $depth ) : '';
        $class_names = $value = '';
        $classes = empty( $item->classes ) ? array() : (array) $item->classes;
        $class_names = join( ' ', apply_filters( 'nav_menu_css_class', array_filter( $classes
    ), $item ) );
        $class_names = ' class="'. esc_attr( $class_names ) . '"';
        $output .= $indent . '<li id="menu-item-'. $item->ID . '"'. $value .
    <class_names . '>';
        $attributes = ! empty( $item->attr_title ) ? ' title="'. esc_attr( $item-
    >attr_title ) . '"' : '';
        $attributes .= ! empty( $item->target ) ? ' target="'. esc_attr( $item->target
    ) . '"' : '';
        $attributes .= ! empty( $item->xfn ) ? ' rel="'. esc_attr( $item->xfn
    ) . '"' : '';
        if($item->object == 'page' && $item->post_parent != 0)
        {
            if( is_front_page() )
            {
                $attributes .= ' href="#post-' . $item->object_id . '"';
            }
            else
            {
                $attributes .= ' href="'.home_url().'/#post-' . $item->object_id . '"';
            }
        }
        else
        {
            $attributes .= ! empty( $item->url ) ? ' href="'. esc_attr( $item->url
    ) . '"' : '';
        }

        $item_output = $args->before;
        $item_output .= '<a'. $attributes . '>';
        $item_output .= $args->link_before . apply_filters( 'the_title', $item->title, $item-
    >ID );
        $item_output .= $args->link_after;
        $item_output .= '</a>';
        $item_output .= $args->after;
        $output .= apply_filters( 'walker_nav_menu_start_el', $item_output, $item, $depth,
    $args, $id );
    }
}

```

Ce walker va permettre de modifier les liens du menu lorsque les items correspondent à des pages enfants, sans modifier ceux des autres items de menu :

1. Si on se trouve sur l'url pointera vers une ancre dont la structure doit être la suivante : **id="post-<?php the_ID(); ?>"**
2. Si on se trouve sur une autre page, les liens de ces items particuliers pointeront sur la page d'accueil et l'ancre permettra d'arriver directement sur la section.

➔ Structure des template-parts des sections

On pourra créer un fichier `template-parts/content-page.php`

```
<section class="one-page" id="post-<?php the_ID(); ?>" itemscope
itemtype="http://schema.org/Article">
  <div class="container">
    <header>
      <h3 itemprop="name"><?php the_title() ?></h3>
    </header>
    <?php if ( has_post_thumbnail() ) : ?>
      <?php the_post_thumbnail( 'thumbnail' ); ?>
    <?php endif; ?>
    <div class="content" itemprop="description">
      <?php the_content(); ?>
    </div>
  </div>
</section>
```

```
<?php if ( $query->have_posts() ) : ?>
  <?php while ( $query->have_posts() ) : $query->the_post(); ?>

    <?php get_template_part( 'template-parts/content' , 'page' ); ?>

  <?php endwhile; ?>
<?php endif; ?>
```

➔ Insertion du walker dans le fichier de fonctions

Nous allons inclure le walker dans le fichier **fonctions.php** (à la suite des autres fonctions)

```
require_once( dirname(__FILE__) . '/inc/customWalker.php' );
```

➔ Modification de l'appel du menu afin de prendre en compte le walker

Dans le fichier **header.php** contenant l'appel du menu :

```
<nav id="navigation" class="menu-header">
  <?php wp_nav_menu( array(
    'theme_location' => 'header' ,
    'container' => '' ,
    'container_id' => '' ,
    'container_class' => '' ,
    'walker'=> new Single_Page_Walker, // du nom de la classe dans customWalker.php
  ) ); ?>
</nav>
```

17-Les champs personnalisés

A) Créer un champ personnalisé

Dans l'interface de saisie d'un article, afficher les champs personnalisés en cliquant sur l'onglet « Option de l'écran » en haut à droite de l'interface

Options de l'écran

Options de l'écran

Un nouvel encart apparaît sous la zone de saisie du contenu

Ajouter un nouveau champ personnalisé :

Nom	Valeur
— Sélectionner —	

Saisissez-en un nouveau

Ajouter un champ personnalisé

Cliquer sur le lien « Saisissez-en un nouveau » :

La zone « Nom » permet de donner un nom à la colonne de table

La zone « Valeur » permet de saisir une information pour cette colonne pour l'article en cours

Puis cliquer sur le bouton « Ajouter un champ personnalisé »

Ajouter un nouveau champ personnalisé :

Nom	Valeur
Un nouveau champ personnalisé	Contenu du nouveau champ personnalisé

Annuler

Ajouter un champ personnalisé

B) Afficher les champs personnalisés

Ouvrir un template part, par exemple template-parts/content.php, et rajouter la fonction permettant d'afficher les champs personnalisés de l'article :

```
<?php if ( has_post_thumbnail() ) : ?>
    <?php the_post_thumbnail( 'paysage' ); ?>
<?php endif; ?>
<div class="content" itemprop="description">
    <?php the_content(); ?>
    <?php the_meta(); ?>
</div>
```

c) Limite fonctionnelle

➔ *Au niveau de l'administration*

Rendez-vous sur un autre article et observez l'encart « Champs personnalisés »

Il est nécessaire de cliquer sur la boîte de sélection pour retrouver le champ créé dans l'article précédent.

➔ *Interface de saisie et Type de champ*

L'interface de saisie est très sommaire et ne permet pas d'encadrer l'utilisateur dans les données attendues.

➔ *Code généré par la fonction the_meta()*

```
<ul class='post-meta'>
  <li><span class='post-meta-key'>Un nouveau champ personnalisé:</span> Contenu du
nouveau champ personnalisé</li>
</ul>
```

Il est possible de personnaliser l'affichage des champs personnalisés, moyennant un effort de programmation en PHP.

18-Le plugin Advanced Custom Field

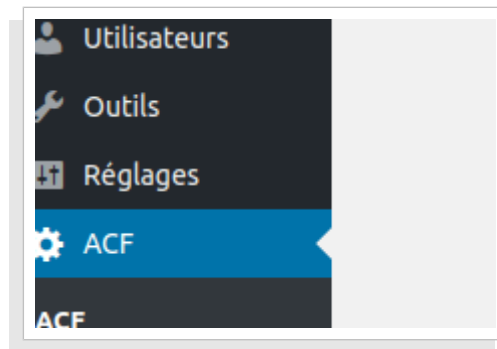
Ce plugin va permettre de créer des champs personnalisés de plusieurs types différents et de créer une interface d'administration robuste.

Sur la plan technique, il s'appuie sur les champs personnalisés de Wordpress et ne pose donc pas de problème en cas d'évolution de la structure de données.

<https://www.advancedcustomfields.com/>

<http://www.geekpress.fr/tutoriel-advanced-custom-fields-debutants/>

Une fois activé, le plugin est accessible via le lien ACF dans le menu,



A) Créer un groupe de champ que nous nommerons : référencement

Nous allons créer 2 zones de saisie supplémentaires dans les articles et les pages afin de personnaliser le <title> et la <meta name= description>.

Un groupe de champ permet d'assigner les champs de saisie de ce groupe à une ou plusieurs ressources Wordpress et d'effectuer des réglages quant à l'affichage de ces champs dans l'interface de saisie des ressources.

Référencement

Position du champ	Titre du champ	Nom du champ	Type de champ
1	Title	title	Texte
2	Description	description	Zone de texte

Faites glisser pour réorganiser

Assigner ce groupe de champs

Règles

Créez une série de règles pour déterminer sur quelles pages d'édition ce groupe de champs sera utilisé

Montrer ce champ quand

Type de publication ▼ est égal à ▼ post

ou

Type de publication ▼ est égal à ▼ page

ou

Ajouter une règle

B) Ajouter des champs dans le groupe

Pour créer un champ, il faut cliquer sur le bouton « Ajouter »

Les champs du groupe s'affichent dans le panneau du haut et peuvent être réorganiser.

Position du champ	Titre du champ	Nom du champ	Type de champ
1	Title	title	Texte
2	Description	description	Zone de texte

Faites glisser pour réorganiser

+ Ajouter

→ Les types de champs

Il y a de nombreuses possibilités, prévues dans la version standard, supplémentaires avec la version PRO et de nombreuses extensions.

Pour le champ title, nous utiliserons le type « texte » qui correspond à une ligne de texte

Pour le champ description, nous utiliserons une « zone de texte » qui correspond à un textarea.

Pour les deux champs, nous appliquerons l'option « Aucun Formatage » et contrôlerons le nombre de caractères, puisque le contenu doit être inséré dans les balises <title> et <meta> respectives.

Type de champ *	Texte
Instructions pour ce champ Instructions pour les auteurs. Affichées lors de la soumission de données.	
Requis ?	<input type="radio"/> Oui <input checked="" type="radio"/> Non
Valeur par défaut Apparaît lors de la création d'un article	
Texte de substitution Apparaît dans la saisie	
Préfixe Apparaît avant la saisie	
Suffixe Apparaît après la saisie	
Formatage Modifie le contenu sur la partie publique du site	Aucun formatage
Nombre de caractères Laisser vide pour aucune limite	200
Logique conditionnelle	<input type="radio"/> Oui <input checked="" type="radio"/> Non
	Fermer le champ

C) Les nouveaux champs dans l'interface de saisie

Modifier l'article

Bonjour tout le monde !

Permalien : <http://local.jdc/wordpress/bonjour-tout-le-monde/>

Title

Description

D) Affichage dans les templates

➔ Désactiver l'ajout automatique du titre dans le fichier de fonctions

```
/*
 * Let WordPress manage the document title.
 * By adding theme support, we declare that this theme does not use a
 * hard-coded <title> tag in the document head, and expect WordPress to
 * provide it for us.
 */
//add_theme_support( 'title-tag' );
```

➔ Exploiter les marqueurs conditionnels et afficher les champs ACF

Dans header.php, utiliser ces nouveaux champs en exploitant les marqueurs conditionnels

https://codex.wordpress.org/fr:Marqueurs_conditionnels

```
<?php if( is_front_page() ) : ?>
    <title><?php bloginfo('name')?></title>
    <meta name="description" content="<?php bloginfo('description')?>">
<?php elseif( is_single() ) : ?>
    <title><?php the_field('title'); ?></title>
    <meta name="description" content="<?php the_field('description'); ?>">
<?php else : ?>
    <title><?php wp_title('') ?></title>
<?php endif; ?>
```


LES CUSTOM POST TYPE

Les custom post types (ou types de contenus personnalisés) permettent de créer du contenu différent (et différencié) de ceux fournis par défaut par WordPress, à savoir les articles ou les pages.

1-Création d'un agenda

Créer un nouveau fichier dans le répertoire inc du thème : inc/customPostAgenda.php

Inclure ce fichier dans le fichier de fonctions

```
...
require_once( dirname(__FILE__) . '/inc/customWalker.php' );
require_once( dirname(__FILE__) . '/inc/customPostAgenda.php' );
```

customPostAgenda.php

```
<?php

add_action( 'init', 'tpw_custom_post_agenda' );

function tpw_custom_post_agenda() {

    // Les différentes dénominations qui seront affichées dans l'administration

    $labels = array(
        'name'          => __( 'Événements' ), // Nom au pluriel
        'singular_name' => __( 'Événement' ), // Nom au singulier
        'menu_name'     => __( 'Agenda' ), // Libellé affiché dans le menu
        // Les différents libellés de l'administration
        'all_items'     => __( 'Tous les événements' ),
        'view_item'     => __( 'Voir les événements' ),
        'add_new_item'  => __( 'Ajouter un nouvel événement' ),
        'add_new'       => __( 'Ajouter' ),
        'edit_item'     => __( 'Editer l\'événement' ),
        'update_item'   => __( 'Modifier l\'événement' ),
        'search_items'  => __( 'Rechercher un événements' ),
        'not_found'     => __( 'Événement non trouvé' ),
        'not_found_in_trash' => __( 'Événement non trouvé dans la corbeille' ),
    );

    $args = array(
        'label'          => __( 'Événements' ),
        'description'    => __( 'Agenda des événements' ),
        'labels'         => $labels, // Le tableau précédent

        // Options disponibles dans l'éditeur : titre, contenu, introduction, image à la une
        'supports'       => array( 'title', 'editor', 'excerpt', 'author', 'thumbnail' ),
        'hierarchical'   => false, // Se comporte comme des articles
        'has_archive'    => true, // Avoir une page pour lister les événements
        'public'         => true, // Visible sur le site public
        'taxonomies'     => array( 'category', 'post_tag' ), // Classer par catégorie,
        // ajouter des mots-clés
    );

    // Enregistrement du custom post type "agenda" et les arguments précédents

    register_post_type( 'agenda', $args );
}
```

Aller dans l'administration : un nouvel item « Agenda » apparaît dans le menu

2-Les templates

Les templates portent en suffixe le nom du custom post type, tel que défini dans **register_post_type('agenda', \$args);**

A) Page d'archive listant les événements

Créer un template archive-**agenda**.php pour afficher plusieurs événements, sur le modèle de de index.php

Créer en template-parts/content-event.php sur le modèle de content.php.

archive-agenda.php

```
<?php get_header(); ?>

<main>
  <div class="site-main">
    <div class="site-main-inner container">

      <?php if ( have_posts() ) : ?>
        <section class="pt-0">
          <h1><?php wp_title('') ?></h1>

          <div class="flex">

            <?php while ( have_posts() ) : the_post(); ?>
              <?php get_template_part( 'template-parts/content' , 'event' ); ?>
            <?php endwhile; ?>

          </div>

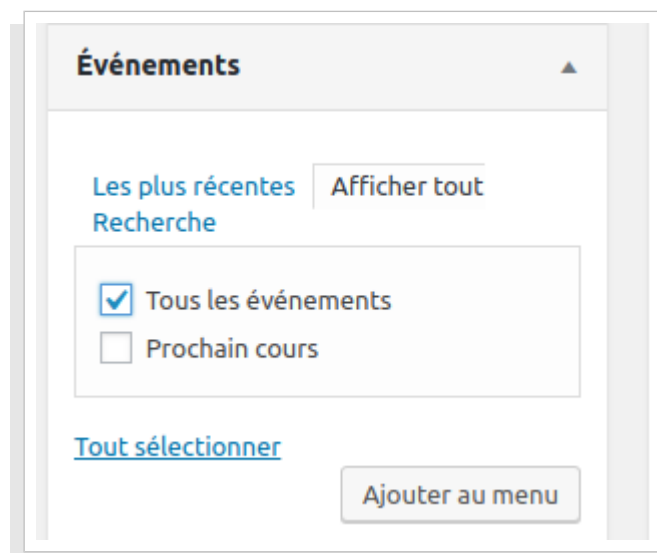
        </section>
      <?php endif; ?>

    </div>
  </div>
</main>

<?php get_footer(); ?>
```

➔ Rajouter un lien vers l'agenda dans le menu

1. Rendez-vous dans Réglages/Permalien et basculer sur l'option « Nom de l'article », puis enregistrer
2. Créer au moins un événement
3. Aller dans le menu et Ajouter « Tous les événements »



Modifier le libelle dans le menu

Il est possible de modifier le libellé d'une page dans le menu (par défaut → le titre)

The screenshot shows a modal window for editing a menu item. The title 'Agenda' is selected. Below it, there are links for navigation: 'Déplacer', 'Un cran vers le haut', 'Sous Blog', and 'Tout en haut'. There is also a field for 'Original' with the value 'Tous les événements'. At the bottom are buttons for 'Retirer' and 'Annuler'.

➔ Requête implicite du template `archive-agenda.php`

La requête implicite du template `archive-agenda.php` affiche les derniers événements publiés, comme la page de blog affiche les derniers articles publiés.

La date de publication de l'événement ne correspond pas à la date de l'événement.

En général, un agenda n'affiche que les événements à venir.

Il est parfois nécessaire de créer une page pour les événements passés afin de faire une rétrospective.

Ce custom-post doit donc être enrichi de champs supplémentaires, tels que la date de l'événement, son lieu, etc.

B) Template single d'un événement

La page d'un événement utilisera le template `single.php`, mais peut être personnalisée avec un template dédié

Créer un template `single-agenda.php` pour afficher un événement

`single-agenda.php`

```
<?php get_header(); ?>

<main>
  <div class="site-main">
    <div class="site-main-inner container">
      <?php while ( have_posts() ) : the_post(); ?>
        <?php get_template_part( 'template-parts/content' , 'event' ); ?>
        <?php endwhile; ?>
      </div>
    </div>
  </main>

  <?php get_footer(); ?>
```

C) Rajouter des champs aux événements avec ACF

1. Créer un nouveau groupe de champs dans ACF appelé : Agenda
2. Monter ce champ quand le type de publication = agenda

➔ *Les champs supplémentaires*

Un événement contient en général les informations suivantes :

Informations	Type de champ	Nom
Titre	Wordpress titre	the_title
Description	Wordpress content	the_content
Date de début	ACF DATE	date_debut
Date de fin	ACF DATE	date_fin
Lieu	ACF Texte	lieu
Adresse	ACF ZONE DE TEXTE	adresse
Éventuellement une carte	AC GOOGLE MAP	map

➔ *Particularité pour les cartes Google*

<https://www.advancedcustomfields.com/resources/google-map/>

Obtention d'une clé API

Il est nécessaire d'**obtenir une clé** :

<https://developers.google.com/maps/documentation/javascript/get-api-key>

Une fois la clé obtenue, ouvrir le fichier de fonctions

```
/*
  ACF API GOOGLE
  https://www.advancedcustomfields.com/resources/google-map/
*/
include_once( ABSPATH . 'wp-admin/includes/plugin.php' );
if ( is_plugin_active('advanced-custom-fields/acf.php') ) {

  add_filter('acf/fields/google_map/api', 'tpw_google_map_api');
  function tpw_google_map_api( $api ){
    $api['key'] = 'ID DE L'API';
    return $api;
  }
}
```

Rajouter des scripts dans les templates

Récupérer le code js sur la page : <https://www.advancedcustomfields.com/resources/google-map/>
et le stocker dans un fichier : js/acf-map.js

Déclarer les scripts dans footer.php

```
<?php wp_footer(); ?>

<script src="https://maps.googleapis.com/maps/api/js?key=Votre clé API"></script>
<script src="<?php echo get_template_directory_uri()?>/js/acf-map.js"></script>

</body>
</html>
```

Rajouter les styles CSS dans la feuille de styles

```
/* =====
CARTE GOOGLE MAP ACF
===== */

.acf-map {
    width: 100%;
    height: 400px;
    border: #ccc solid 1px;
    margin: 20px 0;
}

/* fixes potential theme css conflict */
.acf-map img {
    max-width: inherit !important;
}
```

D) Afficher les nouveaux champs dans les templates agenda

Ouvrir templates-parts/content-event.php (copie de content.php)

On pourra modifier l'itemtype pour adopter la structure de données Event et supprimer les données de publication d'un article.

```
<article id="post-<?php the_ID(); ?>" itemscope itemtype="http://schema.org/Event">
  <header>
    <h2 itemprop="name"><?php the_title() ?></h2>

    <!-- Adresse et lieu -->
    <dl itemprop="location" itemscope itemtype="http://schema.org/Place">
      <dt itemprop="name"><?php the_field('lieu'); ?></dt>
      <dd itemprop="address"><?php the_field('adresse'); ?></dd>
    </dl>

  </header>
  <?php if ( has_post_thumbnail() ) : ?>
    <?php the_post_thumbnail( 'thumbnail' ); ?>
  <?php endif; ?>
  <div class="content" itemprop="description">
    <?php the_content('',true); ?>
  </div>
  <a class="btn" itemprop="url" href="<?php the_permalink(); ?>">
    <?php echo esc_html( 'Lire la suite' ) ?>
  </a>
</article>
```

➔ Formater les dates

```
<?php
// Récupérer la date
$date_debut = get_field('date_debut', false, false);
// Créer un objet php DateTime correspondant à cette date
$date_debut = new DateTime($date_debut);

//Afficher la date selon un format → http://php.net/manual/fr/function.date.php
echo $date_debut->format('Y-m-j');

?>
```

Dans le template-parts/content-event.php, on pourra créer nos objets date avant l'affichage

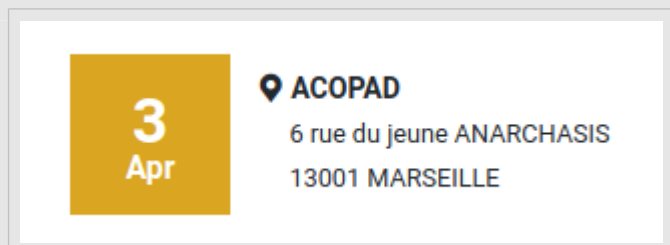
On fournira le format ISO de la date dans l'attribut content

Puis on pourra [styler l'affichage](#)

```
<?php
$date_debut = get_field('date_debut', false, false);
$date_debut = new DateTime($date_debut);
$date_fin = get_field('date_fin', false, false);
$date_fin = new DateTime($date_fin);
?>

<article class="event" id="post-<?php the_ID(); ?>" itemscope
itemtype="http://schema.org/Event">
  <header>
    <p
      class="event-date"
      itemprop="startDate"
      content="<?php echo $date_debut->format('Y-m-j'); ?>"
      <span class="event-date__day">
        <?php echo $date_debut->format('j'); ?>
      </span>
      <span class="event-date__month">
        <?php echo $date_debut->format('M'); ?>
      </span>
    </p>
    <dl class="event-location" itemprop="location" itemscope
    itemtype="http://schema.org/Place">
      <dt itemprop="name"><span class="icon-location" aria-hidden="true"></span><?php
the_field('lieu');?></dt>
      <dd itemprop="address"><?php the_field('adresse');?></dd>
    </dl>
    <h2 itemprop="name" class="clear h4 m-0"><?php the_title() ?></h2>
  </header>
```

```
.event-date {
  float : left;
  margin : 0;
  margin-right : 1rem;
  display: flex;
  flex-direction : column;
  justify-content: center;
  width : 5em;
  height : 5em;
  background : goldenrod;
  color : #fff;
  font-weight : bold;
  text-align: center;
  line-height: 1;
}
.event-date__day {
  font-size : 2em;
}
.event-location {
  overflow: hidden;
  margin : 0.5rem;
  font-size : 0.9em;
}
.event-location dt {
  font-weight: bold;
}
.event-location dd {
  margin-left : 1.5em;
  font-size : 0.9em;
}
```



E) Afficher uniquement les événements à venir

Les événements à venir sont ceux dont la date de départ est supérieure ou égale à **la date du jour**.

Il est donc nécessaire de créer une requête personnalisée selon ce critère de date, lequel correspond à un champ personnalisé (champ meta).

Il est possible de créer un template de page introduisant cette requête, ou d'utiliser le template d'archive agenda.

Dans le fichier archive-agenda.php

```
<?php

$today = date('Ymd');

$args = array (
    'post_type' => 'agenda', //Sélection des custom-post agenda
    'meta_key'   => 'date_debut', // Critère de sélection sur un champ meta
    'orderby'    => 'meta_value_num', // Ordre selon la meta_key
    'order'      => 'ASC',
    // Critère de sélection sur la meta-key : date_debut ≥ $today
    'meta_query' => array(
        array(
            'key'       => 'date_debut',
            'compare'    => '>=',
            'value'      => $today,
        ),
    ),
);

$query = new WP_Query( $args );

get_header(); ?>

<main>
<div class="site-main">
    <div class="site-main-inner container">

        <?php if ( $query->have_posts() ) : ?>
            <section class="pt-0">
                <h1><?php wp_title('') ?></h1>

                <div class="flex">

                    <?php while ( $query->have_posts() ) : $query->the_post(); ?>
                        <?php get_template_part( 'template-parts/content' , 'event' ); ?>
                    <?php endwhile; ?>

                </div>

            </section>
        <?php endif; ?>

    </div>
</div>
</main>

<?php get_footer(); ?>
```

F) Créer une page pour afficher les événements passés

Il suffira de créer une page depuis l'administration et de lui affecter une template de page.

La date de début doit être inférieure à la date du jour.

En copiant archive-agenda sur agenda-passe.php

```
<?php

/* Template Name: Événements passés */

$today = date('Ymd');

$args = array (
    'post_type' => 'agenda',
    'meta_key'   => 'date_debut',
    'orderby'    => 'meta_value_num',
    'order'      => 'ASC',
    'meta_query' => array(
        array(
            'key'       => 'date_debut',
            'compare'    => '<',
            'value'      => $today,
        ),
    ),
);

$query = new WP_Query( $args );

get_header(); ?>
```

➔ *Faire un lien vers les pages*

Dans le template archive-agenda.php, on pourra rajouter un lien vers la page des événements passés

```
<a class="btn" href="<?php echo esc_url( get_permalink(ID DE LA PAGE) ); ?>"><?php echo
esc_html( 'Voir les événements passés' ); ?></a>
```

➔ *Faire un lien vers une page d'archive*

Dans le template de page agenda-passe.php, on pourra rajouter un lien vers la page des événements à venir

```
<a class="btn" href="<?php echo get_post_type_archive_link( 'agenda' ); ?>"><?php echo
esc_html( 'Voir les événements à venir' ); ?></a>
```


3-Les mu-plugins

A) Des plugins simplifiés

Un **MU Plugin** ou **Must-Use Plugin** est un plugin simplifié se composant d'un seul fichier PHP placés dans un répertoire spécifique : **/wp-content/mu-plugins/**

Un **Must-Use plugin** est une solution simple pour rendre les Custom Post Types et autres morceaux de code accessibles en cas de changement de thème.

Il ne s'agit pas d'un plugin à part entière, lesquels contiennent plus de fichiers stockés dans un répertoire dédié.

Les MU plugins apparaissent dans la liste des plugins installés **indispensables** mais ne peuvent pas être désactivés, sauf en supprimant le fichier du plugin dans le répertoire.

B) Faire de l'agenda un mu-plugin

L'agenda précédemment créé ne sera disponible que sur le thème tp-webdesigner.

Afin de le rendre accessible pour tous les thèmes, nous allons déplacer le fichier :

wp-content/themes/tpw-webdesigner/inc/customPostAgenda.php

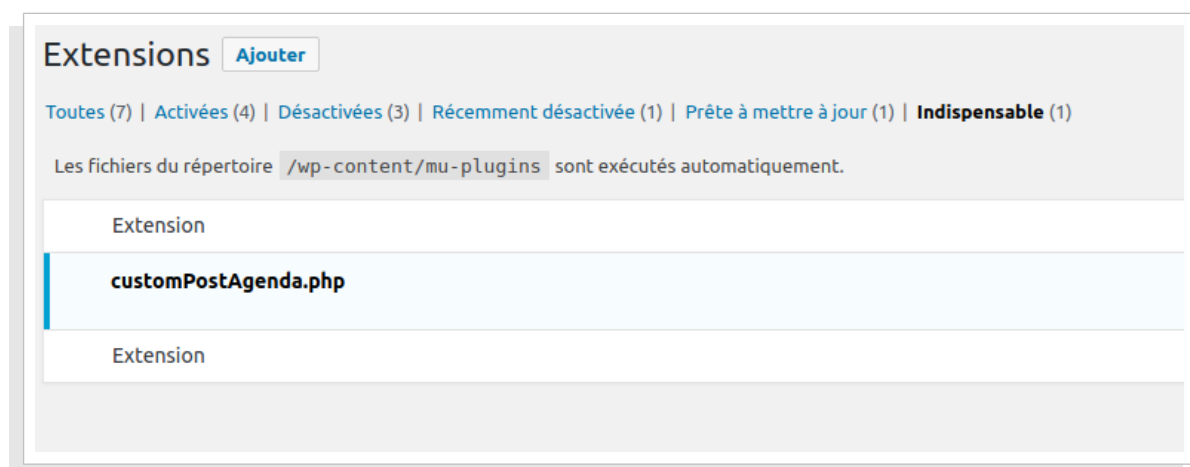
dans

/wp-content/mu-plugins/customPostAgenda.php

NE PAS OUBLIER DE SUPPRIMER OU DE COMMENTER LA LIGNE QUI INCLUT LE FICHIER DANS LE FICHIER DE FONCTIONS DU THÈME

```
require_once( dirname(__FILE__) . '/inc/customWalker.php' );

/* Fichier déplacé dans /wp-content/mu-plugins/ */
//require_once( dirname(__FILE__) . '/inc/customPostAgenda.php' );
```



C) Faire du menu onepage un plugin

Dans la même optique, on pourra déplacer le fichier inc/customWalker.php dans le répertoire, en le renommant de manière plus pertinente :

/wp-content/mu-plugins/customWalkerOnePage.php

D) Faire un plugin simplifié pour un menu déroulant au clic

(Se référer au TP du menu déroulant pour le script et la CSS)

Dans **/wp-content/mu-plugins/**, créer un fichier **dropdownMenu.php**

➔ Enregistrer le script *dropdown_js*

Avec dépendance jquery

```
/* =====
    MENU DROPDOWN
    Insérer le script dans wp_footer()
=====*/

add_action( 'wp_enqueue_scripts', 'dropdown_js' );
function dropdown_js()
{
    wp_register_script(
        'jQueryDropdownMenu',
        get_template_directory_uri() . '/js/jqueryDropdown.js', // URL du script
        array('jquery'), // Dépendance avec d'autres script
        '1.0', // Numéro de version du script
        true // Chargement via wp_footer
    );

    wp_enqueue_script( 'jQueryDropdownMenu' );
}
```

Sans dépendance jQuery (le dropdown est géré en js)

```
/* =====
    MENU DROPDOWN JS
    Insérer le script dans wp_footer()
=====*/

add_action( 'wp_enqueue_scripts', 'dropdown_js' );
function dropdown_js()
{
    wp_register_script(
        'jsDropdownMenu',
        get_template_directory_uri() . '/js/jsDropdown.js', // URL du script
        array(), // Dépendance avec d'autres script
        '1.0', // Numéro de version du script
        true // Chargement via wp_footer (par défaut : chargement dans le head avec wp_head)
    );
    wp_enqueue_script( 'jsDropdownMenu' );
}
```

➔ *Hook pour rajouter un filtre dans le menu*

Un « hook » est un mécanisme permettant d'effectuer une action précise à un moment particulier et sont classés en 2 catégorie :

1. Les **actions** basées sur des événements et déjà introduites avec la fonction **add_action()**
2. Les **filtres** permettent de modifier le texte affiché à l'écran ou enregistré dans la BD, utilisées une fonction **add_filter()**.

Nous allons ajout des attributs aria aux **liens ouvrant un sous-menu au clic**, si l'item correspondant (balise li) a la classe **menu-item-has-children**

Il s'agit donc d'intercepter **la fonction qui génère les liens du menu :**
nav_menu_link_attributes()

https://codex.wordpress.org/Plugin_API/Filter_Reference/nav_menu_link_attributes

```
/* =====
MENU DROPDOWN
https://codex.wordpress.org/Plugin_API/Filter_Reference/nav_menu_link_attributes
===== */
add_filter('nav_menu_link_attributes', 'tpw_add_aria_dropdown', 10,3);
function tpw_add_aria_dropdown($atts, $item, $args) {

    if( in_array( 'menu-item-has-children', $item -> classes ) )
    {
        $atts['role'] = 'button';
        $atts['aria-haspopup'] = 'true';
        $atts['aria-expanded'] = 'false';
    }
    return $atts;
}
```

Nota :

Ce filtre ne fonctionnera pas avec le customWalkerOnePage mais l'ajout des attributs aria y est prévu dans la dernière version.

➔ *Insérer un lien personnalisé (dropdown) dans le menu WORDPRESS*

PARTIR D'UN THÈME EXISTANT

1-Les starter thèmes

Le **starter thème** est une base de départ permettant de coder un thème sur mesure, tout en conservant la maîtrise du code.

Il en existe du plus simple au plus complet.

Exemple de thème starter simple et minimaliste :

<http://html5blank.com/>

Exemple de framework très complet, ce dernier est basé sur bootstrap 4

<https://understrap.com/>

2-Les thèmes enfants

Un thème enfant dans WordPress est un thème qui hérite du code et des fonctionnalités d'un autre thème WordPress, alors appelé le thème parent. Il offre la possibilité de **personnaliser un thème existant sans pour autant y apporter de modifications directes**. Il s'agit donc d'une **surcouche** sur laquelle travailler plutôt que de modifier le thème parent lui-même.

Un thème enfant est composé d'un répertoire installé dans le répertoire wp-content/themes.

Le répertoire du thème enfant sera préférentiellement nommé avec **le nom du thème parent** suivi de **-child** (par exemple : twentyseventeen-child)

Deux fichiers sont obligatoires :

A) style.css

La ligne **Template** correspond au nom du répertoire du thème parent.

(Attention! Ne pas mettre d'espace entre le **paramètre** et :)

```
/*
Theme Name: Twenty Seventeen Child
Description: Twenty Seventeen Child Theme
Author: Votre nom
Template: twentyfifteen
Version: 1.0.0
*/
```

B) functions.php

La fonction suivante chargera la feuille de style du thème parent (**get_template_directory_uri()** est le chemin du **thème parent**)

```
add_action( 'wp_enqueue_scripts', 'theme_enqueue_styles' );
function theme_enqueue_styles() {
    wp_enqueue_style( 'parent-style', get_template_directory_uri() . '/style.css' );
}
```

Le fichier fonctions du thème enfant est chargé en plus de celui du thème parent.

c) Les templates

Le thème enfant peut écraser n'importe quel fichier du thème parent : il suffit simplement d'inclure un fichier du même nom dans le répertoire du thème enfant, et il va écraser le fichier équivalent dans le répertoire du thème parent au chargement du site.

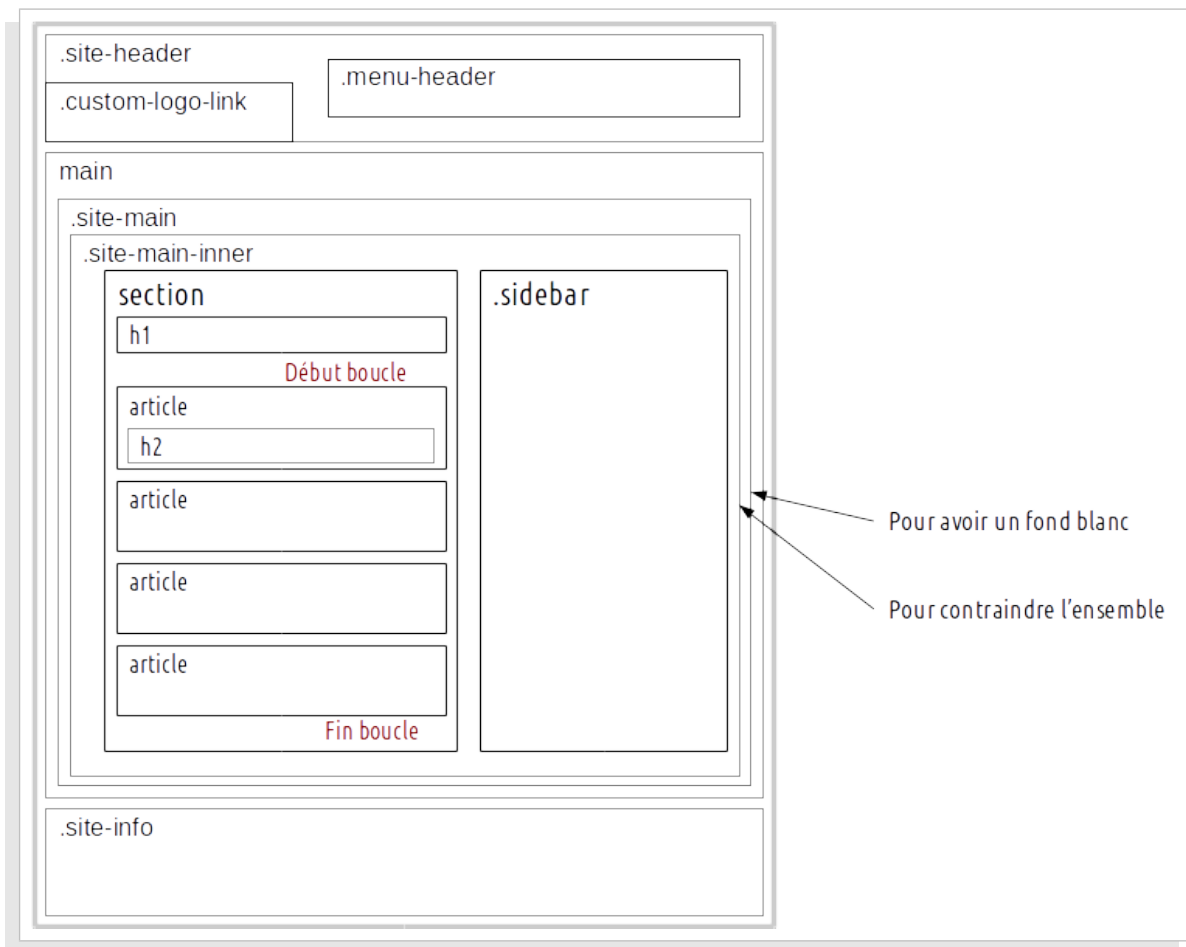
<https://bonjourwp.com/definitions-theme-parent-theme-enfant/>

<https://wpformation.com/theme-enfant-wordpress/>

Exemple de thème enfant tout prêt : <https://github.com/understrap/understrap-child>

ANNEXES

1-Mockup page d'archives



2-Mockup single



3-Mockup page d'accueil onepage

