IES GRANCAPITÁN
CÓRDOBA

Horas de Libre Configuración
2º DESARROLLO DE APLICACIONES WEB

WordPress

UNIDAD 6

PARTE 2- Estructura de archivos y carpetas de WordPress

ÍNDICE DE CONTENIDOS (1/3)

IES GRAN CAPITÁN
CÓRDOBA

ÍNDICE DE CONTENIDOS (2/3)

IES GRAN CAPITÁN
CÓRDOBA

ÍNDICE DE CONTENIDOS (3/3)

IES GRAN CAPITÁN
CÓRDOBA

08

FUNCIONES PARA LOS TEXTOS

IES GRAN CAPITÁN
CÓRDOBA

8. FUNCIONES PARA TEXTOS

Hay dos funciones de WordPress recurrentes en la mayoría de los temas: **__()** y **_e()**.

```
<?php _e( $text, $domain ); ?>
<?php __( $text, $domain ); ?>
```

Argumentos

$text: el texto a traducir.

$domain: el nombre clave del archivo en el que se encuentran las cadenas de traducción.

```
<?php _e('Nothing Found', 'twentytwenty'); ?>
```

IES GRANCAPITÁN
CÓRDOBA

09

LAS RUTAS EN LAS URL

IES GRAN CAPITÁN
CÓRDOBA

9. LAS RUTAS EN LAS URL

site_url(): devuelve la dirección del sitio web como variable.

Ejemplos: http://www.misitio.com o http://www.misitio.com/carpeta-Wordpress

home_url(): devuelve la dirección de la raíz del sitio de WordPress como variable.

Ejemplo: http://www.misitio.com

admin_url(): devuelve la dirección de la administración en forma de variable.

Ejemplo: http://www.misitio.com/wp-admin

includes_url(): devuelve la ruta del directorio wp-includes como una variable.

Ejemplo: http://www.misitio.com/wp-includes

9. LAS RUTAS EN LAS URL

content_url(): devuelve la ruta de la carpeta wp-content como una variable.
Ejemplo: http://www.misitio.com/wp-content

plugins_url(): devuelve la ruta de la carpeta de complementos como una variable.
Ejemplo: http://www.misitio.com/wp-content/plugins

theme_url(): devuelve la ruta de la carpeta de temas como una variable.
Ejemplo: http://www.misitio.com/wp-content/themes

wp_upload_dir(): devuelve la ruta de la carpeta de cargas como una variable.
Ejemplo: http://www.misitio.com/wp-content/uploads

9. LAS RUTAS EN LAS URL

get_template_directory_uri(): en el caso de un tema hijo, devuelve la URL de la carpeta del tema padre; de lo contrario, devuelve la carpeta del tema activo.

get_stylesheet_directory_uri(): en el caso de un tema hijo, devuelve la URL de la carpeta del archivo style.css hijo; de lo contrario, devuelve la carpeta del archivo **style.css** del tema activo.

Las funciones **bloginfo($param)** y **get_blog info($param)** también pueden devolver una ruta URL. Consulte la siguiente sección para obtener detalles de estas funciones

10

LA FUNCIÓN
wp_nav_menu()

IES GRAN CAPITÁN
CÓRDOBA

10. LA FUNCIÓN wp_nav_menu()

Permite mostrar cualquier menú creado en la administración **Apariencia - Menús**, en archivos de WordPress. Normalmente se encuentra en el archivo **header.php**, pero puede variar según el tema.

```php
<?php
$defaults = array(
 'menu' => '',
 'menu_class' => 'menu',
 'menu_id' => '',
 'container' => 'div',
 'container_class' => '',
 'container_id' => '',
 'fallback_cb' => 'wp_page_menu',
 'before' => '',
 'after' => '',
 'link_before' => '',
 'link_after' => '',
 'echo' => true,
 'depth' => 0,
 'walker' => '',
 'theme_location' => ''
 'items_wrap' => '<ul id="%1$s"
class="%2$s">%3$s</ul>',
 'item_spacing' =>'preserve',
);
wp_nav_menu( $defaults );
?>
```

IES GRANCAPITÁN
CÓRDOBA

10. LA FUNCIÓN wp_nav_menu()

Argumentos

menu: acepta el nombre o el id. de un menú previamente creado o existente en la administración, a través de Apariencia - Menús.

Ejemplo: 'menu' => 'el-nombre-de-mi-menu'

menu_class: acepta un nombre de clase para la etiqueta HTML <ul>, que incluye etiquetas HTML <li>.

Ejemplo: 'menu_class' => 'clase-del-menu'

```
<ul class='clase-del-menu'>
   <li><a href='http://mienlace.com'>nombre de mi enlace</a></li>
   <li><a href='http://mienlace2.com'>nombre de mi enlace2</a></li>
</ul>
```

IES GRANCAPITÁN CÓRDOBA

10. LA FUNCIÓN wp_nav_menu()

Argumentos

menu_id: acepta un nombre de id. para la etiqueta HTML <ul> que engloba las etiquetas HTML <li>.

Ejemplo: 'menu_id' => 'id-del-menu'

```
<ul id='id-del-menu'>
   <li><a href='http://mienlace.com'>nombre de mi enlace</a></li>
   <li><a href='http://mienlace2.com'>nombre de mi enlace2</a></li>
</ul>
```

container: acepta el nombre de la etiqueta HTML que contiene el menú. Por defecto: . Para no tener etiqueta, establezca el valor en false.

Ejemplo: 'container' => 'span'

```
<span>
     <ul>
        <li><a href='http://mienlace.com'>nombre de mi enlace</a></li>
        <li><a href='http://mienlace2.com'>nombre de mi enlace2</a></li>
     </ul>
</span>
```

IES GRANCAPITÁN CÓRDOBA

10. LA FUNCIÓN wp_nav_menu()

Argumentos

container_class: acepta un nombre de clase para la etiqueta que contiene el menú.

Ejemplo: 'container_class' => 'clase-del-contenedor'

```
<span class='class-del-contenedor'>
   <ul>
       <li><a href='http://mienlace.com'>nombre de mi enlace</a></li>
       <li><a href='http://mienlace2.com'>nombre de mi enlace2</a></li>
   </ul>
```

fallback_cb: acepta un nombre, devuelto si el menú no existe. Utilice false para desactivar este argumento, por defecto: wp_page_menu.

Ejemplo: 'fallback_cb' => 'false'

IES GRAN CAPITÁN
CÓRDOBA

10. LA FUNCIÓN wp_nav_menu()

Argumentos

before: acepta un texto que se muestra antes de la etiqueta HTML <a href> (enlace de cada pestaña).

Ejemplo: 'before' => '>'

```
<ul>
    <li>><a href='http://mienlace.com'>nombre de mi enlace</a></li>
    <li><a href='http://mienlace2.com'>nombre de mi enlace2</a></li>
</ul>
```

after: acepta un texto que se muestra después de la etiqueta HTML <a href> (enlace de cada pestaña).

Ejemplo: 'after' => '<'

```
<ul>
   <li><a href='http://mienlace.com'>nombre de mi enlace</a><</li>
   <li><a href='http://mienlace2.com'>nombre de mi enlace2</a></li>
</ul>
```

IES GRAN CAPITÁN
CÓRDOBA

10. LA FUNCIÓN wp_nav_menu()

Argumentos

link_before: acepta un texto que se muestra antes del texto del enlace en la etiqueta HTML <a href> (enlace de cada pestaña). Por lo tanto, se puede hacer clic en el texto.

Ejemplo: 'link_before' => '>'

```
<ul>
   <li><a href='http://mienlace.com'>> nombre de mi enlace</a></li>
   <li><a href='http://mienlace2.com'>> nombre de mi enlace2</a></li>
</ul>
```

link_after: link_after: acepta un texto que se muestra después del texto del enlace en la etiqueta HTML <a href> (enlace de cada pestaña). Por tanto, se puede hacer clic en el texto.

Ejemplo: 'link_after' => '<'

```
<ul>
   <li><a href='http://mienlace.com'>nombre de mi enlace <</a></li>
   <li><a href='http://mienlace2.com'>nombre de mi enlace2 <</a></li>
</ul>
```

IES GRANCAPITÁN CÓRDOBA

10. LA FUNCIÓN wp_nav_menu()

Argumentos

echo: acepta true o false. Le permite mostrar o utilizar el menú como variable. Por defecto: true.

Ejemplo: 'echo' => '0' o 'echo' => 'false'

depth: acepta un número utilizado para administrar los niveles del menú.

Por defecto: **0**, que muestra el menú completo. **1** desactiva todos los submenús a partir del nivel 1.

En el caso de los submenús, **2** desactiva todos los submenús a partir del nivel 2 y abandona los submenús del nivel 1.

-1 muestra todas las pestañas al mismo nivel.

10. LA FUNCIÓN wp_nav_menu()

Argumentos

```
<ul>
   <li><a href='http://mienlace.com'>nombre de mi enlace</a></li>
     <ul>
       <li>
            <a href='http://mienlace-nivel1.com'>nombre de mi enlace nivel1</a>
       </li>
     </ul>
   <li><a href='http://mienlace2.com'>nombre de mi enlace2</a></li>
</ul>
```

IES GRANCAPITÁN
CÓRDOBA

10. LA FUNCIÓN wp_nav_menu()

Argumentos

walker: acepta un objeto, no una cadena de caracteres. Le permite personalizar completamente su menú. Por defecto: new Walker_Nav_Menu.

Ejemplo: 'walker' => 'new mi-objeto'

theme_location: acepta el nombre de la ubicación. Este campo le permite guardar un menú en una ubicación específica.

En administración, la ubicación aparece en **Apariencia - Menús - Gestionar ubicaciones**. Primero se debe definir usando la función register_nav_menus() (que verá en detalle al crear menús personalizados en el capítulo Personalizar el sitio con el archivo functions.php - sección Crear ubicaciones para los menús).

Ejemplo: 'menú' => 'nombre-de-ubicación-de-mi-menú'

https://developer.wordpress.org/reference/functions/wp_nav_menu/

11

HOOKS (FILTROS Y ACCIONES)

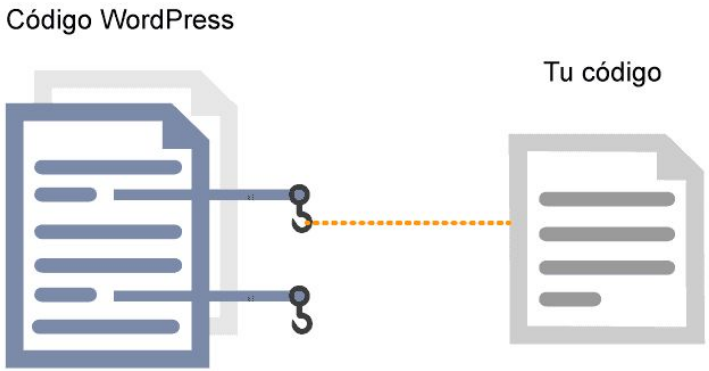
IES GRAN CAPITÁN
CÓRDOBA

11. HOOKS (FILTROS Y ACCIONES)

HOOKS

A través de los Hooks en WordPress puedes cambiar, añadir, o eliminar funcionalidades que vienen con WordPress y hacerlo de manera ordenada y sin perder los cambios cuando se actualice WordPress.

El término *"hook"* se puede traducir como gancho ⟶ enganchamos código adicional al código del core de WordPress



IES GRAN CAPITÁN
CÓRDOBA

11. HOOKS (FILTROS Y ACCIONES)

HOOKS

Tipos: **acciones** y **filtros**.

1. Acciones (Action Hooks)

Ejecutan una acción pero que no devuelven ningún valor. Básicamente es agregar funcionalidad en un punto determinado del código de WordPress.

```
<?php add_action($hook, $function_name, $priority, $accepted_args);?>
```

Se define los Actions Hooks con la función *do_action()*, sin embargo para usar los Hooks usaremos la función *add_action()*.

https://www.webempresa.com/blog/hooks-wordpress-como-utilizarlos-guia-rapida.html

IES GRAN CAPITÁN
CÓRDOBA

11. HOOKS (FILTROS Y ACCIONES)

Argumentos de add_action

$hook: argumento obligatorio. Acepta un nombre de acción (aquí está la lista: https://codex.wordpress.org/Plugin_API/Action_Reference, https://adambrown.info/p/wp_hooks) o el nombre de una acción creada en una extensión o un tema, gracias a la función **do_action().**

$function_name: argumento obligatorio. Acepta el nombre de la función donde se desarrolla la acción.

$priority: argumento opcional. Se utiliza para especificar el orden en el que se debe realizar la acción. Por defecto: 10. Cuanto menor sea el número, antes se llevará a cabo la acción. Las acciones con la misma prioridad se ejecutan en el orden en que aparecen.

$accepted_args: argumento opcional. Se usa para especificar el número de argumentos que pasan en la función **$function_name.**

11. HOOKS (FILTROS Y ACCIONES)

Ejemplos

El siguiente código agrega un archivo javascript a nuestro sitio web:

```
add_action("wp_enqueue_scripts", "web_insertar_js");

function web insertar js(){
          wp_register_script('miscript', get_stylesheet_directory_uri().
'/js/script.js', array('jquery'), '1', true );
          wp_enqueue_script('miscript');   }
```

En el código anterior hemos usado el Hook wp_enqueue_scripts para añadir un archivo javascript, como puedes ver en la función add_action se hace referencia a la función web_insertar_js en donde está toda la lógica para añadir el archivo **script.js.**

IES GRAN CAPITÁN
CÓRDOBA

11. HOOKS (FILTROS Y ACCIONES)

HOOKS

Tipos: **acciones** y **filtros**.

2. Filtros (Filter Hooks)

Permiten ejecutar código en un determinado punto, sin embargo en este caso la función tendrá obligatoriamente que devolver un valor modificado.

```
<?php add_filter($hook, $function_name, $priority, $accepted_args);?>
```

Se define los “Fiters Hooks” con la función *apply_filter()*, sin embargo para usar los Hooks usaremos la función *add_filter()*.

IES GRAN CAPITÁN
CÓRDOBA

11. HOOKS (FILTROS Y ACCIONES)

Argumentos de add_filter

$hook: argumento obligatorio. Acepta un nombre de filtro (aquí está la lista: http://codex.wordpress.org/Plugin_API/Filter_Reference, http://adambrown.info/p/wp_hooks) o el nombre de una acción creada en una extensión o un tema, con la función apply_filter().

$function_name: argumento obligatorio. Acepta el nombre de la función donde se realiza el filtrado.

$priority: argumento opcional. Se utiliza para especificar el orden en el que se debe ejecutar el filtro. Por defecto: 10. Cuanto menor sea el número, antes se ejecutará el filtro. Los filtros con la misma prioridad se ejecutan en el orden en que aparecen.

$accepted_args: argumento opcional. Se usa para especificar el número de argumentos que pasan en la función **$function_name.**

11. HOOKS (FILTROS Y ACCIONES)

Ejemplos

El siguiente código modifica el enlace de *leer más* en el listado de entradas::

```
function web_modificar_leer_mas() { return ' Leer más '; }

add_filter( 'the_content_more_link', 'web_modificar_leer_mas' );
```

En el código anterior hemos usado el Hook the_content_more_link, el cual hace referencia a la función web_modificar_leer_mas dentro de esta función podemos hacer las modificaciones HTML al enlace de leer más y devolverlo modificado.

IES GRAN CAPITÁN
CÓRDOBA

11. HOOKS (FILTROS Y ACCIONES)

ENCONTRAR HOOKS EN WORDPRESS

Buscar en el código de WordPress e ir viendo qué Hooks tengo disponibles para determinada funcionalidad, básicamente buscaría las funciones do_action y apply_filter, que como habíamos visto son las funciones de Actions Hooks y Filter Hooks.

Una forma alterna es buscar los Hooks por orden alfabético, para esto existe el sitio web WordPress a2z, que clasifica los Hooks para su fácil localización.

. ENLACES RECOMENDADOS

https://brandominus.com/blog/creatividad/funciones-para-wordpress-hasta-por-las-orejas/

IES GRAN CAPITÁN
CÓRDOBA