college CDI

Development of a dynamic web site (DWD)

Class 06

The function.php file

We already have created the *functions.php* and you already know it is a mandatory file for all WordPress themes. In this file, WordPress functionalities can be activated and our own personal custom functions can be coded. To do so, the *hooks system*, which is a very important concept of the CMS, will be used to connect to various WordPress features.

function.php purposes

Several things can be accomplished using this file. As you will gradually discover, many of WordPress functionalities cannot be activated from the dashboard and will therefore have to be defined, configured and activated in the *function.php* file.

Previously, we have used this file to activate the feature image as well as the automatic title tag generation as you can see in the following example.

```
<?php
add_theme_support( 'post-thumbnails' );
add_theme_support( 'title-tag' );
```

Among others, the following actions can be performed using *function.php* file :

- Activate feature image function;
- declare menus and widgets zones ;
- declare CSS stylesheets and JS scripts files;
- declare new types of publications and taxonomies ;
- declare custom functions;
- manage AJAX requests;
- reformat URLs;
- customize certain extensions settings;
- customize administration's interface;
- create API Rest paths;

The hooks system

The hook system allows you to use a specific features, WordpPress functionalities, at various moments while using the CMS. It makes it possible to interact with the theme and the extensions so their default behaviours may be modified without touching their native codes.

Important

It is very important to avoid modifying the codes of the various WordPress components. Anyways, regular updates of the application or of extensions would overwrite your modifications.

The hooks

Hooks are functions declared allowing to interact with WordPress, themes and/or extensions at key moments of their execution. There are two different types of hooks : ***actions*** and ***filters***.

Actions :
Actions define key moments when our own custom functions are triggered allowing additional features.

Filters :
Filters are used to retrieve values at various given moments so they can be modified or updated.

Examples :

An action may be added which calculates the number of words contained in an article at the moment it is saved.

An extension such as *Yoast* can use a filter to retrieve and modify a title upon pages requests for better natural SEO.

Example of the use of hooks in function.php file

Hooks are everywhere in WordPress. Extensions operations, for instance, entirely are based on hooks. It is the same with custom themes creation. The functions.php file will mainly use hooks such as in the following example.

```
<?php
function my_project_custom_function() {
            remove_menu_page( 'tools.php' );
   remove_menu_page( 'edit-comments.php' );
}
add_action( 'admin_menu', 'my_project_custom_function' );
```

Explanation :

This custom function named *my_project_remove_menu_page()*, removes useless elements from WordPress admin dashboard.

The hook is called using the function *add_action()*. The key moment is *admin_menu*. which means that the function is called when WordPress gets ready to display the admin menu.

So, in resume, the function tells WordPress to launch the custom function which removes certain elements from the admin menu prior to the launch of the default menu.

Important

Be careful to use original descriptive custom functions names to make sure there are no possible conflict with existing native functions. A good habit is to use your project's name's initials as a prefix to your functions names.

Loading external scripts and stylesheets

It would be tempting to link our scripts and stylesheets in the header, like it's usually done, being careful to place the links to custom files last. Although, it cannot be done this way with WordPress because the theme and various extensions are declaring their own scripts and stylesheets. WordPress automatically loads and manages all these files in order using the function *wp_head()* which was placed in *header.php*. Therefore, these scripts and stylesheets will have to be declared in *functions.php*.

Declaring scripts and stylesheets in functions.php

Declaration is done using ***add_action('wp_enqueue_scripts', 'your_function');***. See below.

```
<?php
function myproject_register_assets() {          // Custom function name

wp_enqueue_script(                    // To declare jQuery CDN
        'jquery',
        'https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js',
        false,
        '3.4.1',
        true
);

wp_enqueue_script(                    // To declare external JS
        'myproject',
        get_template_directory_uri() . '/js/script.js',
        array( 'jquery' ),
        '1.0',
        true
);

wp_enqueue_style(                     // To declare a stylesheet at the root
        'myproject',
        get_stylesheet_uri(),
        array(),
        '1.0'
);

wp_enqueue_style(                     // To declare another CSS file
        'myproject',
        get_template_directory_uri() . '/css/main.css',
        array(),
        '1.0'
);
}
add_action( 'myproject_enqueue_scripts', 'myproject_register_assets' );
```

wp_enqueue_script()

Allows to declare a JS file in the page.

wp_enqueue_style()

Allows to declare a CSS file.

Parameters to add to the function :

First parameter (the handle) :
Name given to your file (e.g. : your project's name).

Second parameter (file address) :
Using *get_template_directory_uri()* supplies the path to the theme. To declare a CSS file in the theme's root, the use *get_stylesheet_uri()* alone.

Array() :
The array is used to have WordPress scripts and stylesheets declared first.

Version number :
the version number is used to indicate the version number of the files. When you update the file, simply increase the version number (e.g. : 1.1) so the browser's files in cache won't be used.

Boolean :
A boolean is used last in scripts declaration. If *true*, the script is placed at the end of the page, if *false* the script is placed at the beginning of the page.

jQuery :
In order to use the latest version of jQuery, since WordPress does it another way for older Explorer browsers, we have declared another version.

Tip

Using the Network tab of the browser's Inspector will let you see all the requests made and the total time needed to complete them. Optimization is an important aspect when it comes to developing themes.

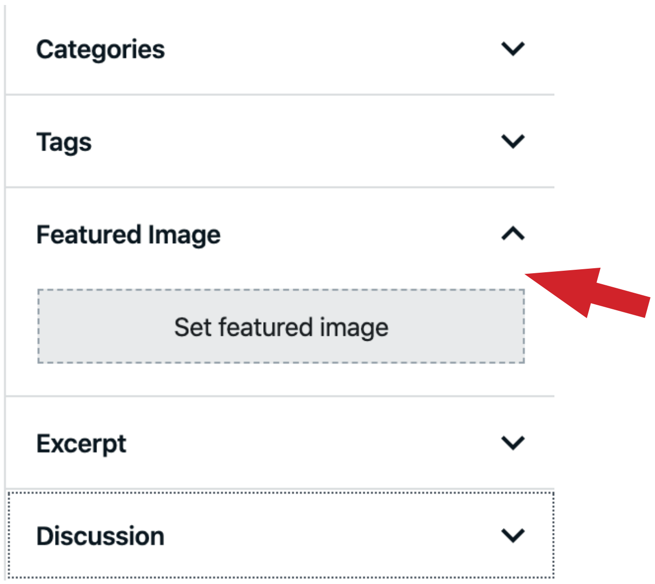
For more information :
https://developer.wordpress.org/reference/functions/wp_enqueue_style/
https://developer.wordpress.org/reference/functions/wp_enqueue_script/

The featured image

As you already know, the feature image function makes it possible to add an image to an article.

Activate featured image in functions.php

The function ***add_theme_support('post-thumbnail')*** activates the featured image functionality in functions.php (we already have done this). This actually adds the feature image metabox to the admin interface.



Displaying the featured image in a template

To display the featured image in any of the templates, the function to be used is ***the_post_thumbnail()*** which needs to be placed within the WordPress loop :

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

        <h1><?php the_title(); ?></h1>
        <?php the_post_thumbnail(); ?>

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

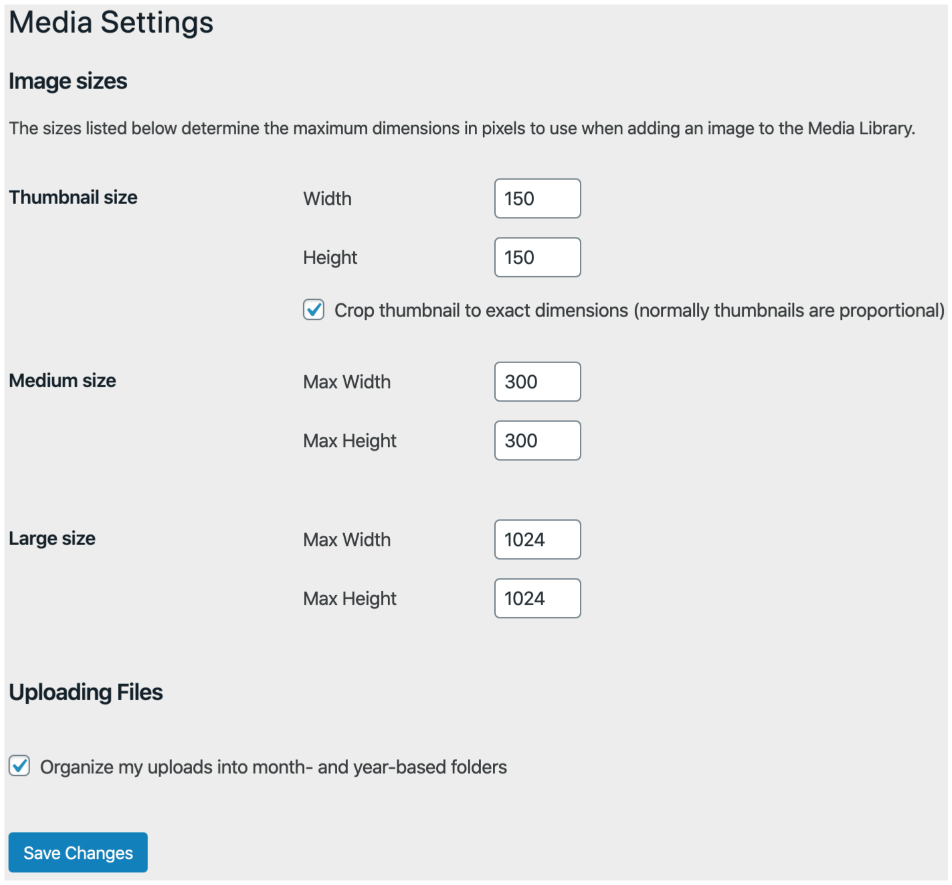
<img> tag

The image tag is automatically created by the system which also adds different sizes of images to be used in different situations.

Configuring images sizes

When an image is imported to WordPress, they're automatically treated so different image sizes are available to be used as featured images, in articles and templates for different devices.

Images sizes can be customized from the dashboard, selection **Settings > Medias**. It is generally suggested to set medium size to 1200px, and large size to 1920px.



Media Settings

Image sizes

The sizes listed below determine the maximum dimensions in pixels to use when adding an image to the Media Library.

Thumbnail size	Width	150
	Height	150
	✓ Crop thumbnail to exact dimensions (normally thumbnails are proportional)	
Medium size	Max Width	300
	Max Height	300
Large size	Max Width	1024
	Max Height	1024

Uploading Files

✓ Organize my uploads into month- and year-based folders

Important

Any changes made to the media settings does not apply to any of the images previously imported.

You may then choose to resize these images manually, or to use a plugin such as *Regenate Thumbnails* to batch resize them.

https://wordpress.org/plugins/regenerate-thumbnails/

Declaring new images sizes

If the three sizes of images are not enough for you, it is possible to declare some more in *functions.php* :

set_post_thumbnail_size()

This function allows to define the size of the featured image which will be used as default size by *the_post_thumbnail()*.

```
set_post_thumbnail_size( 2000, 400, true );
```

First parameter :
Defines the width of the image.

Second parameter :
Defines the height of the image (Writing « 0 » wont limit the height).

Third parameter :
When set to *true*, it allows the image to be cropped to fit the precise dimensions.

Adding new images sizes

In order to use these additional image formats you have just created into posts or articles, the function ***add_ image_size()*** will be used in the file *function.php*.

```
add_image_size( 'products', 800, 600, false );
```

First parameter :
Name of your custom size (avoid using the names used by WordPress).

Second parameter :
Defines the width of the image.

Third parameter :
Defines the height of the image.

Fourth parameter :
Activate *(true)* or deactivate *(false)* the crop option.

Important

Adding custom images sizes will result in more versions of all imported images.

Using custom images sizes in templates

To call a special image size in a template, all that is needed is to use its name in the parameters of the function *the_post_thumbnail()*:

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

Using custom images sizes in articles

When the image bloc option is used in the admin dashboard, a drop-down menu allows you to choose the size you want to use for a given image. But your custom sizes won't show-up in this menu.

In fact, you won't need to bother about this menu if you have created custom sizes as the proper size will automatically be selected by the system depending on the situation. You can then simply set all images to Large to avoid using the original version image (usually much heavier).

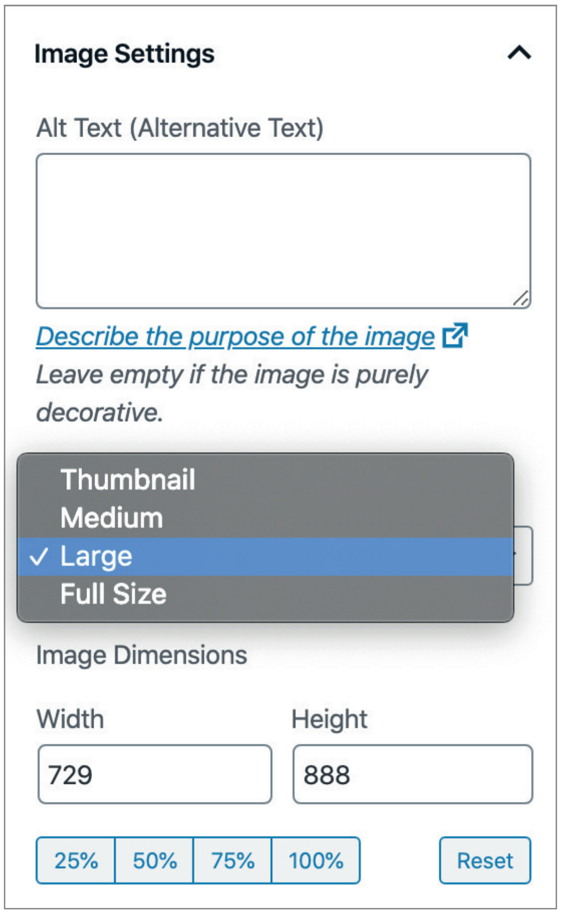


Image Settings

Alt Text (Alternative Text)

Describe the purpose of the image

Leave empty if the image is purely decorative.

- Thumbnail
- Medium
- ✓ Large
- Full Size

Image Dimensions

Width: 729

Height: 888

Pagination of archives

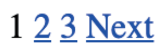
Pagination allows navigation through archive pages. It is rather simple to integrate using predefined ready to use functions. To add pagination, you simply need to add the function after the loop of *archive.php* for a blog or *front-page.php* file for a normal web site. It could also be placed in the footer.

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

<h1 class="site__heading">My archive page</h1>

<div class="site__blog">
    <main>
        <?php if( have_posts() ) : while( have_posts() ) : the_post(); ?>
                <!-- Content of the article -->
        <?php endwhile; endif; ?>

        <?php posts_nav_link();?>
    </main>

    <aside>
        <?php dynamic_sidebar( 'blog' ); ?>
    </aside>
</div>

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

<?php posts_nav_link();?>
Shows a link saying « Next page ».

<a href="https://wp.local/blog/">« Previous page</a> -
<a href="https://wp.local/blog/page/3/">Next page »</a>
Shows link saying « Previous page - Next page ».

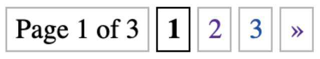
<?php the_posts_pagination(); ?>
Shows a title saying « Posts navigation » which can be hidden using CSS followed by the page number and a « Next » option as shown below.


1 2 3 Next

Pagination using a plugin

You can use a plugin called *WP-PageNavi* which produces a better looking pagination.

Download the plugin https://wordpress.org/plugins/wp-pagenavi/ and use the following snippet instead of the previous ones : **<?php wp_pagenavi(); ?>**


Page 1 of 3 | 1 | 2 | 3 | »