college CDI

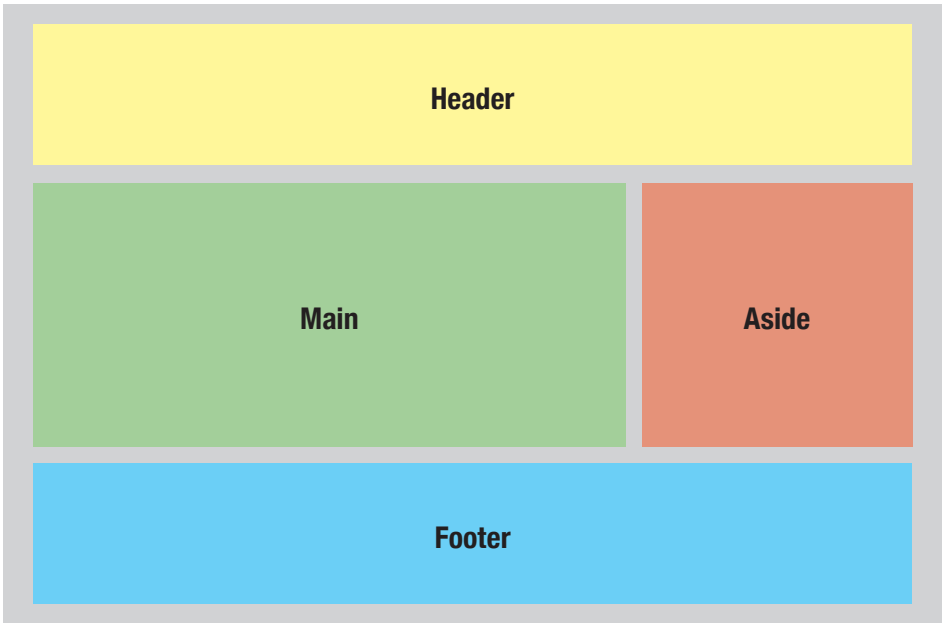
Development of a dynamic web site (DWD)

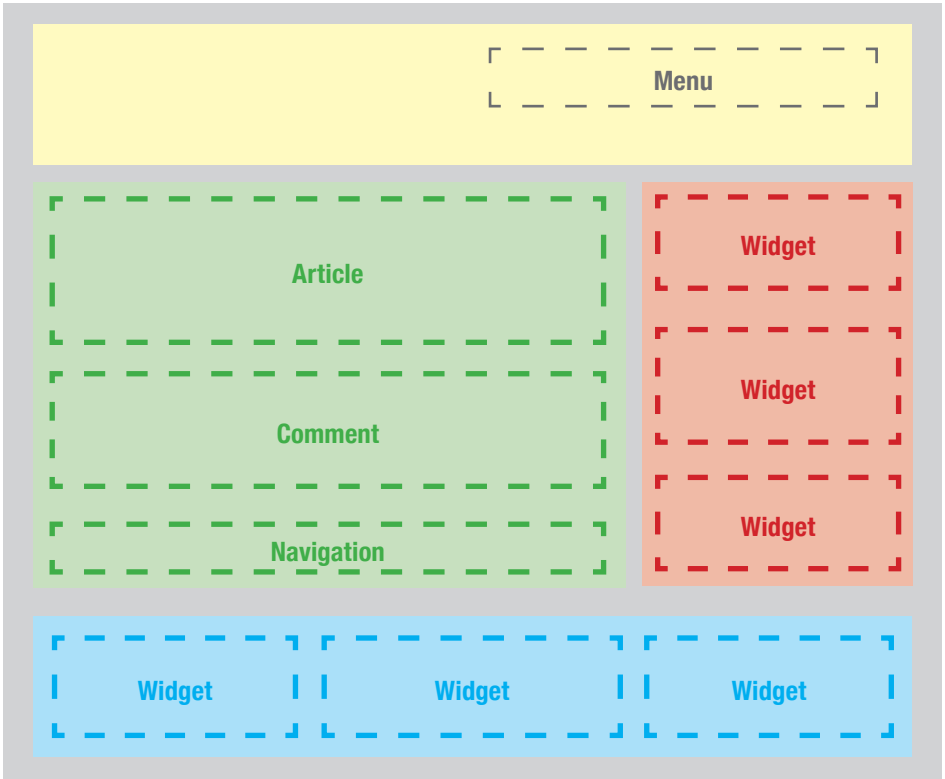
Class 02

Developing a custom WordPress theme

Typical zones and components of web pages

Web sites have come to quite similar ways to fragment and position content elements. You can see below how most web sites divide the pages in zones containing various components. Although zones should always remain the same from one page to another, some elements may vary.





Minimal components of a custom theme

Now has come the time to create the very basic files of a new custom WordPress theme. You may use a code editor or Notepad (or simple text) as you wish. And an image of 880×660 pixels will also be needed.

Step 1 : Creating the theme's folder

In **Wp-Content/Themes**, create a folder for the new theme you will be creating. Place in the folder the following files you will need to create :

index.php
This is the first (and for now the only) page of the theme.

style.css
Used to style the page, but also to declare the theme to WordPress, the CSS must start with a multilines comment containing various information as you can see in the example below.

```
/*
Theme Name: my_project
Theme URI: https://www.myproject.com
Author: Jean G. Turgeon
Author URI: https://www.monamijean.com
Description: My first custom WordPress Theme
Requires at least: WordPress 5.0
Version: 1.0
*/
```

functions.php
Used to code every functionalities. The following codes activates the Features image function (post-thumbnails) and the automatic page title generation (title-tag).

```
<?php
// Activation of post-thumbnails
add_theme_support( 'post-thumbnails' );

// Automatically adds the title in the web site's header
add_theme_support( 'title-tag' );
```

screenshot.png
Used a preview of the theme which must be 880×660 pixels.

Step 2 : Activate the custom theme

In the dashboard, in the **Appearance/Themes** tab, click the *Activate* for the custom theme. From now on, the content of the pages will be positioned and styled by the theme.

Step 3 : Test the theme

Making sure something is written in the *index.php* file such as a title, open the page in MAMP to make sure it works. If the page's content is showing : it works!

Integrating header and footer

The header and the footer are elements that usually never change from one page to another. To avoid having to copy and paste codes in every pages, these will be coded in external files which will also include WordPress codes.

Step 1 : Create the header.php file

This file contains the start of the index file's source code plus a few PHP codes including the function displaying the header.

```
<!DOCTYPE html>
<html <?php language_attributes(); ?>>
<head>
<meta charset="<?php bloginfo('charset'); ?>">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-
to-fit=no"/>
<?php wp_head(); ?>
</head>
<body <?php body_class(); ?>>
<?php wp_body_open(); ?>
```

Step 2 : Create the footer.php file

It contains the end of the index file including the PHP function displaying the footer.

```
<?php wp_footer(); ?>
</body>
</html>
```

Step 3 : Modify the index.php file and test it

Your *index.php* file will then simply be reduced to two PHP scripts calling *header.php* and *footer.php* and the text content of the page (the inspector should now show much more source code then before).

```
<?php get_header(); ?>
<h1>Hello world!</h1>
<?php get_footer(); ?>
```

Explanation of the added WordPress codes

language_attributes()
Added to the <body> tag, it automatically defines the document's language based on the customization made in the dashboard settings under tabs **General/Site language**.

bloginfo('charset')
Defines the default characters encoding (default is UTF-8).

wp_head()
This is a very important function as it allows to load your theme, plugins and styles. For instance, this is what writes the <title> into the pages.

body_class()
This allows to retrieve CSS classes names corresponding the page visited.

In the browser's inspector

Using the inspector, note the classes that were added to <body>.

```
<!DOCTYPE html>
<html lang="en-EN">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-
to-fit=no"/>

        <!-- Many lines of codes -->

</head>
<body class="home blog logged-in admin-bar no-customize-support">
```

home
Because this is the home page.

blog
Because the home page should display the most recent posts (this can be customized).

logged-in
Because we are a connected user.

admin-bar
To display the admin bar at the top of the page.

Integrating a logo in the header

Step 1 : Create an image folder

In the theme's root, create a folder named ***img*** and place an image you will be using as a logo into it.

Step 2 : Code the image in the header

In the file *header.php*, add the proper code in order to display the logo.

```
<!DOCTYPE html>
<html <?php language_attributes(); ?>>
<head>
<meta charset="<?php bloginfo('charset'); ?>">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-
to-fit=no"/>
<?php wp_head(); ?>
</head>
<body <?php body_class(); ?>>
<?php wp_body_open(); ?>

<header class="header">
<a href="<?php echo home_url( '/' ); ?>">
<img src="<?php echo get_template_directory_uri(); ?>/img/logo.png" >
</a>
</header>
```

home_url()

This is the default address of the web page, the root of the site : index.php.

get_template_uri()

This is used to get the absolute address of the image
(e.g. : http://wp.local/wp-content/themes/my_project/img/logo.png).