Day 1 - Tuesday, 1st December 2020
Fundamental WordPress

Section 1 - Intro

Step 1 - Intro

Section 2 - Domain Name And Hosting

Step 2 - What is a domain name and host ?

1. Domain name is your website name, such as google.com

2. Hosting name

Your computer request to access disney.com to another computer, then that computer will send back all the logo, images, text, files, and show the disney.com

Step 3 - Getting Your Domain Name and Host

1. HostGator

- Choose your plan / package
- Choose to register a new domain or I already own this domain (Register)
- Fill the domain and extension
- Domain privacy protection to protect our phone number, email (because it is available) (uncheck is okay)
- Choose a hosting plan : better choose 1 month / 1-year package
- Enter your billing info : all of your data
- Add additional service :
    - SSL Certificate is the security for our website, to keep our visitor data and build confidence in our website (actually it is free, but not sure what is the different between free and must pay)
    - SiteLock Essentials to protect our website from hackers with help from SiteLock. (Highly recommended, but we can do this by our own code for free, so we may uncheck it)

- Get professional email and more from Microsoft Office 365 : we can use Google for the service (better)
- Site Backup to back up our hard work
- HostGator SEO Tools to improve our search raking and increase website visitors

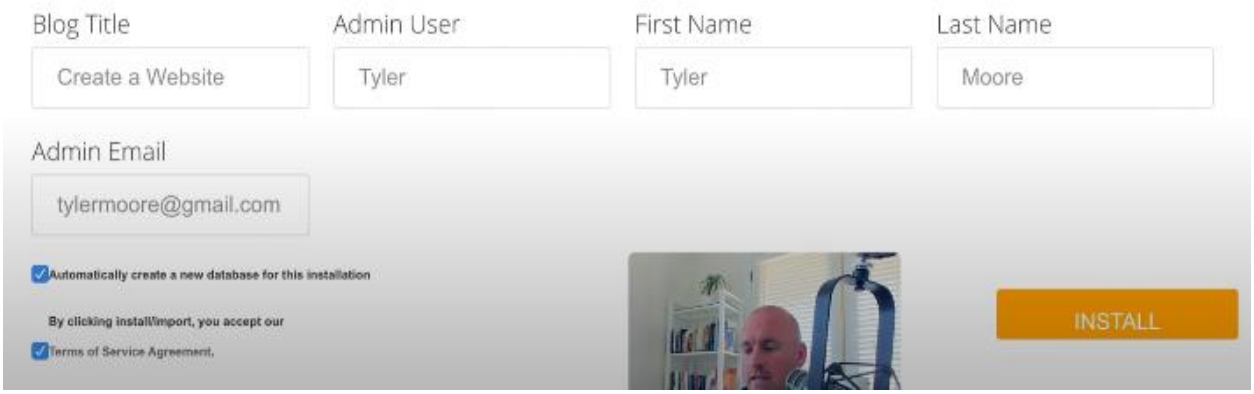
- Add some discount coupon
- Check it now to start your hosting

Section 3 - Setting Up Your Website

Step 4 - Install WordPress

1. Login in to HostGator, go to Marketplace, Related Services, and Once-Click Installs (WordPress logo)

2. Choose WordPress, fill the domain name and make the directory = blank, press next. Fill the textbox like below. Then, Install. (Wait around 30 minutes)



Step 5 - Login

1. Access the "website-name/wp-admin" and login use the Intallation username through your cPanel (HostGator).

Step 6 - Change Password

1. After login, go to Users menu to change your password, scroll down and click generate password, put your new password and click update profile.

Step 7 - Delete Plugins

1. Sometimes, when we install a new WordPress, there are some plugins has been installed, but we dont need that. Therefore, we need to uninstall unecessary plugins.
2. Click all the plugins, then deactivate it, and apply.
3. Select all the plugins again, press delete, and apply.

Step 8 - Update WordPress

1. Use to protect our website from hackers.
2. If there is a new version, you may click Update.

Step 9 - Check Permalinks

1. Click Create a website and visit website.
2. We may see some trash link like below.

What we want is a clear domain, like airbnb.com/invite (invite)

3. What should we do ?

Go to Settings, choose post name, and save changes. (Much better)

4. Go to visit website, click hello world, and the link will be clear now.

Step 10 - Change title and Tagline

1. We want to change our website title and tagline like below.

Create a Website — Just another WordPress site

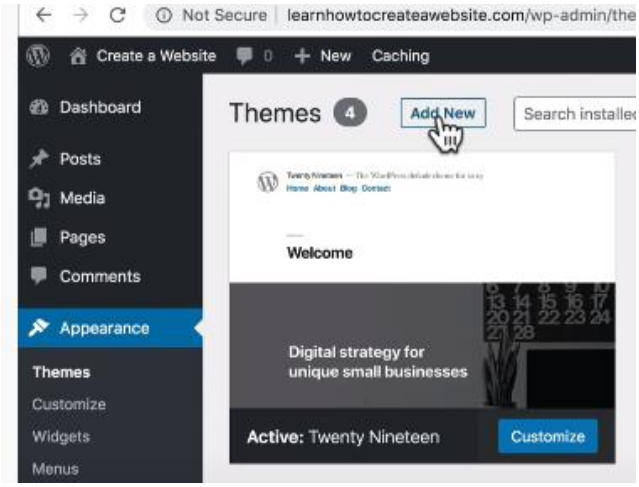
2. Click Customize, go to Site Identity.

3. Change title and tagline.

4. Click Publish.

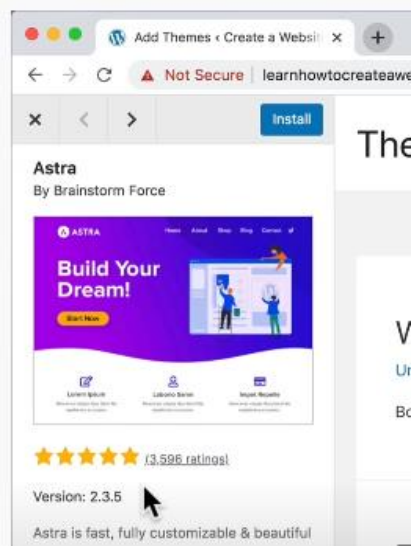
Step 11 - Install Astra Theme

1. Go to our website name, Dashboard, Appearance, Themes.



2. Click Add New, we may search some templates, such as Astra.

3. Click Detail and Install and Activate.

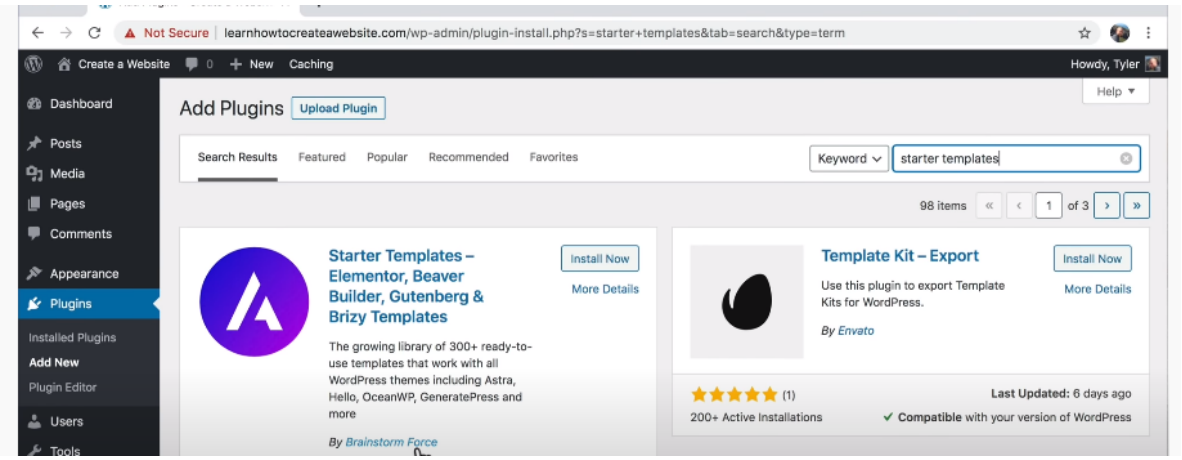


4. Click Visit Site and the theme will be changed.

Step 12 - Install Starter Templates Plugin

1. Click create a website, Dashboard, Plugins, and click Add New.

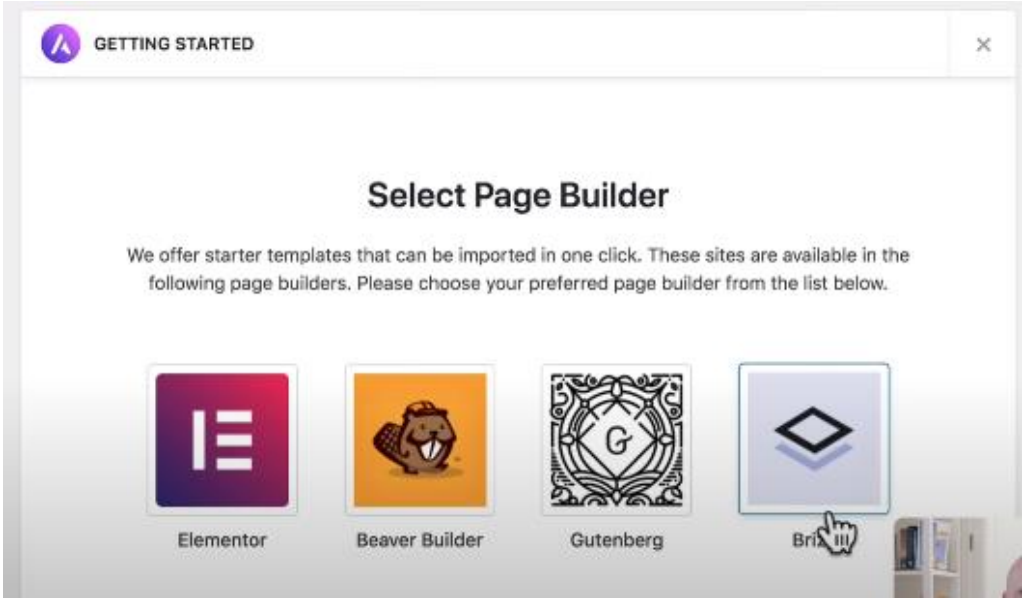
2. Search 'starter templates'. Click Activate.



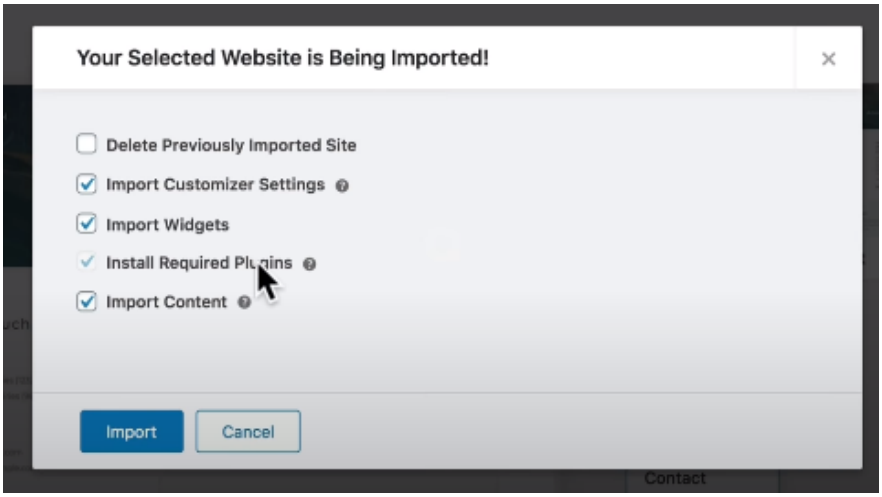
Section 4 - Building Our Website

Step 13 - Choose Your Design

1. Click Appearance and 'Name of Plugins' which is starter templates.



2. Select Elementor is better. Click on free (drop down)
3. Choopse Mountain and we can choose one of the template pages (services, etc)
4. Click Import Complete Templates and click Import.



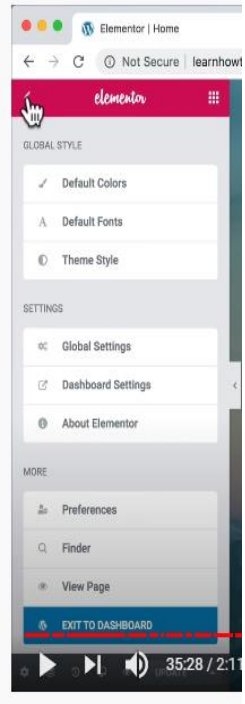
5. All the defaul template has been made but we still need to customize it based on our requirements.

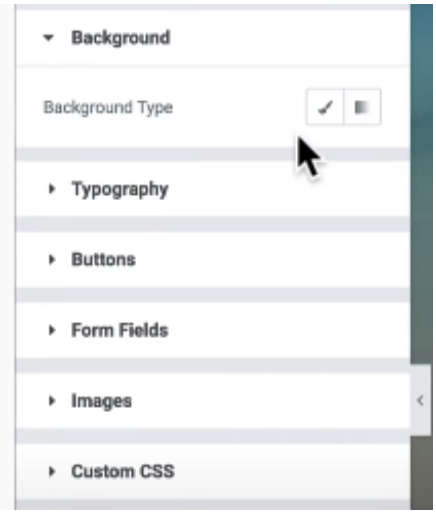
Step 14 - Change Style

1. We may change all the style that we will use in our website. (font family, font size, etc)

2. Click Edit Elementor. (Upper part of our website in Menu Bar)

3. Click Theme Styles.





4. We may change our h1 and h2 in Typography. Go to h1 or h2 and click Typography button.

5. We also may change the button style by choosing Button Menu.

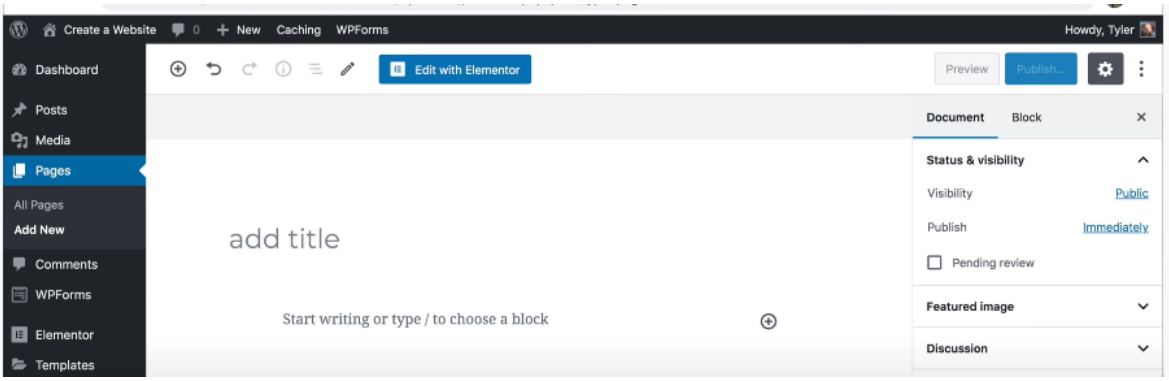
Step 15 - Add Pages

1. Click on Create a Website, go to Dashboad

2. Click on Pages

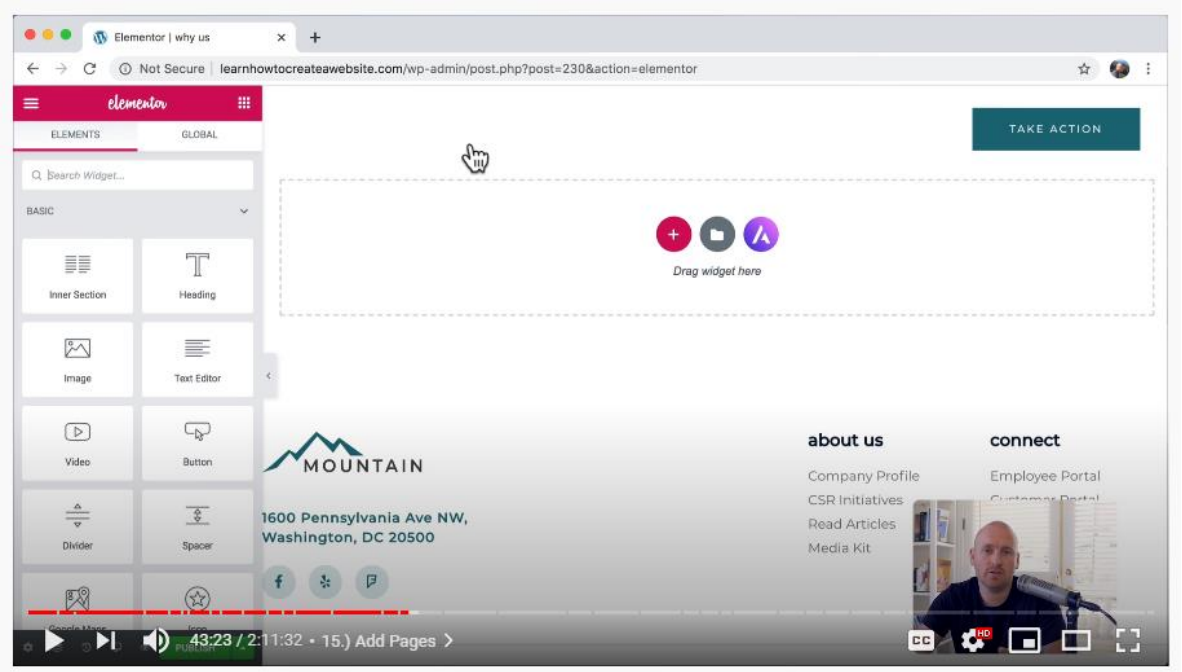
3. If you want to delete a page, just click Trash button on the pages that you want to delete. Go to Trash and click Delete Permanently.

4. Click Add New to add a new page.



5. Click Transparent Header, Enabled on the right button part.

6. Click Edit Elementor to edit the pages.



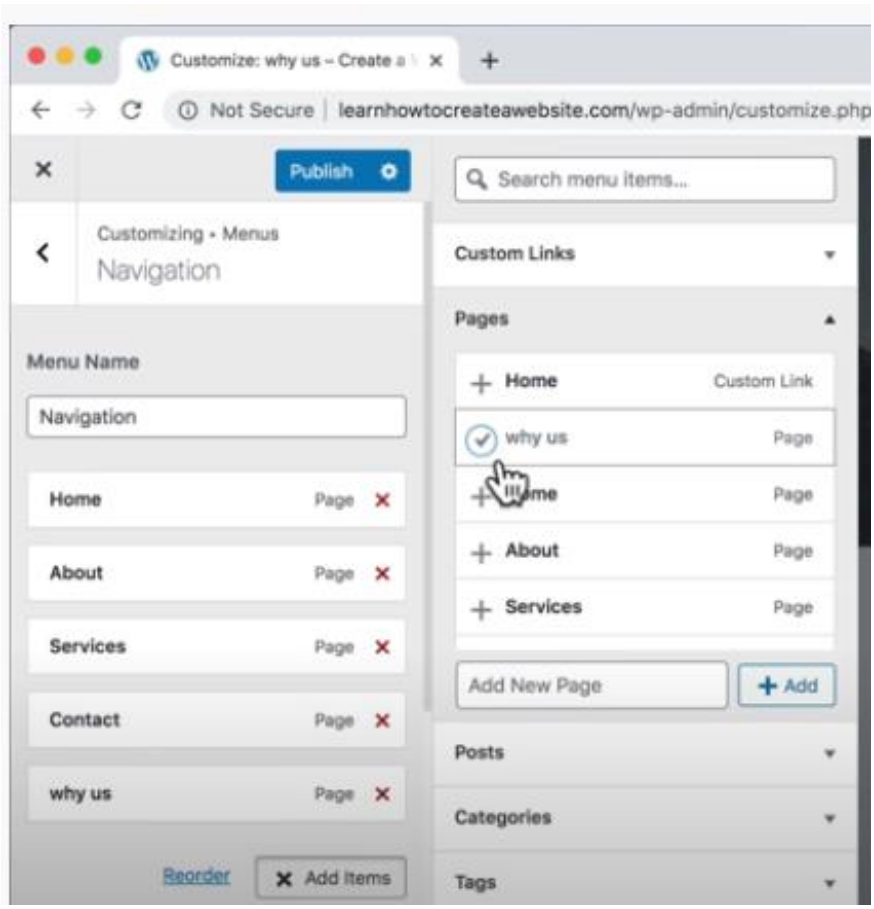
7. Press on A button to choose a new template. (Press Free to get a free template)

8. Click on the template and Import the Template. We can delete a part of the tempalte (like header or footer by clicking x button and we just get the necesseary part from the template)

9. Click Publish if you finish edit your page.

Step 16 - Navigation

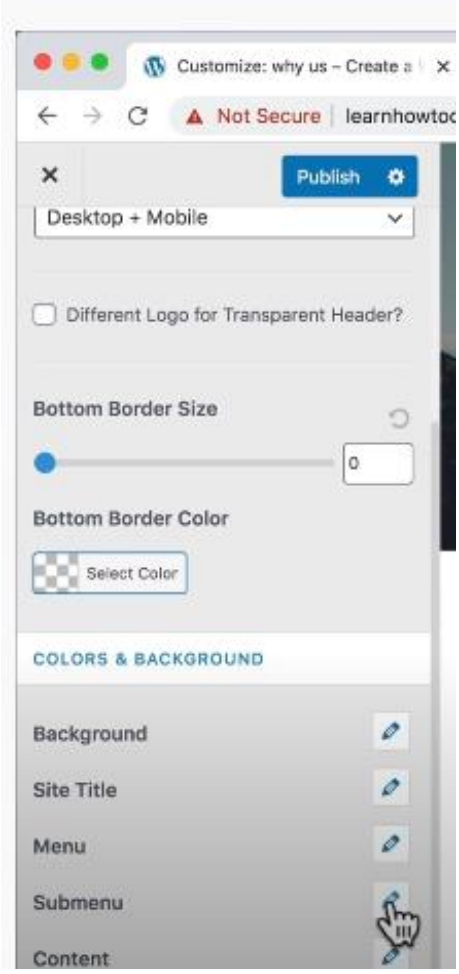
1. Click Customize, Menus, Navigation, and click on Add Items.



2. We can rename the Title of Navigation by clicking the dropdown button (v).

3. Drag it to rearrange the menu or maybe put it below on the other link to make a child navigation.

4. To change the color (anything), go to Header, and Transparent Header. Scroll down and change the Background.



5. Click on Primary Header to change the layout style.

6. Click on Primary Menu to change the last item in Menu, such as Search, Button, etc. Create Contact (remove the old one) and click it.

Step 17 - Logo

1. Go to Header, Site Identity, and Site Logo.
2. To make a logo you can use Photopes and to get a free image ou can use Iconfinder.com
3. Upload your images and import it to Photopes.
4. Export as png
5. Go to our website, click Change Logo.
6. Drag and drop the icon. Upload.

Step 18 - Favicon

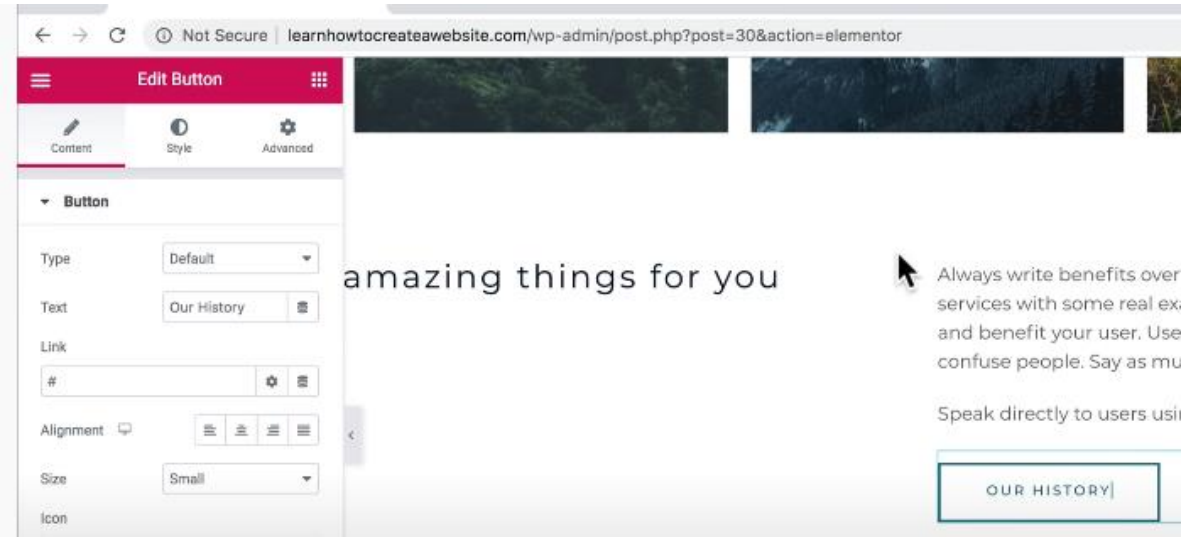
1. Add a small logo into our title pages. (Make a small logo, 512 x 512)

2. Select side icon and upload your small logo, Select. Click Publish.

Section 5 - Design Your Pages

Step 19 - How It Works (Editing Your Websites)

1. Edit in Elementor: click on some part that you want to change and edit it on the elementor (left side)



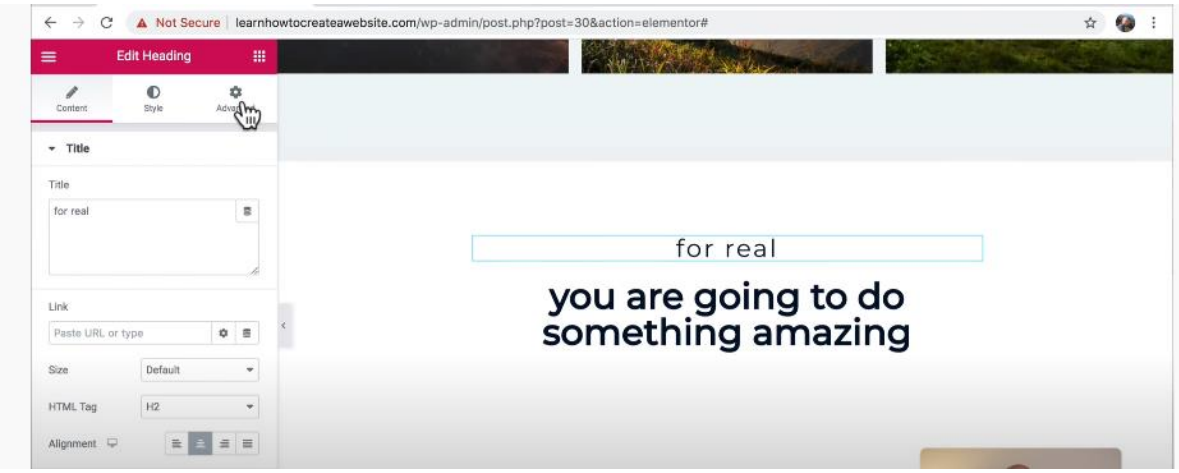
2. Edit widgets or elements (headings, paragraphs, buttons). You may press Ctrl Z if you want to undo it.

Step 20 - Design Like Apple

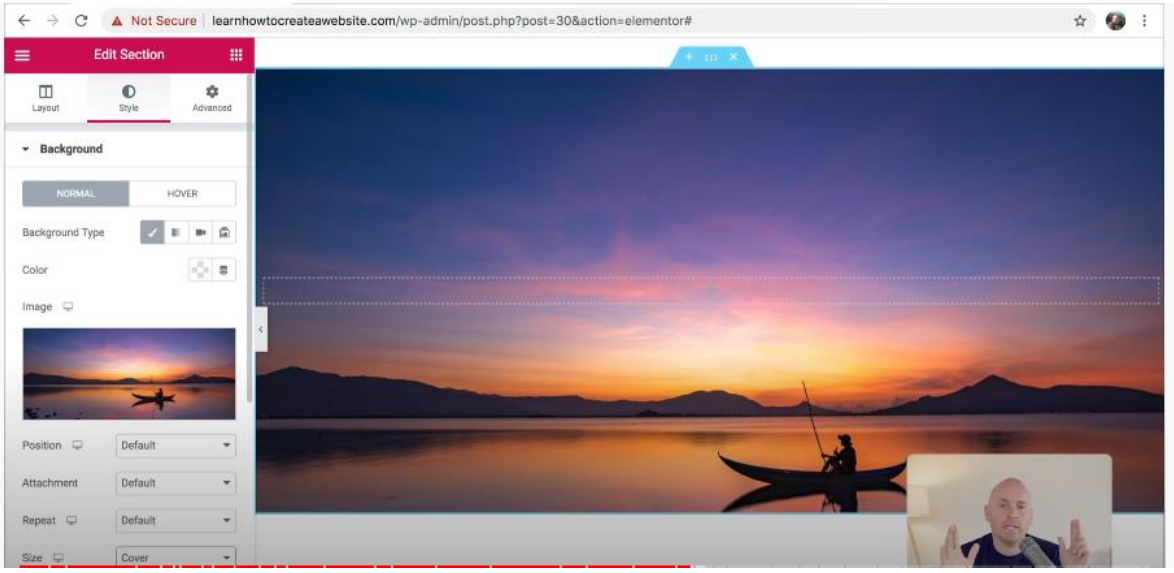
1. We may also add some widgets, just go to the end of the webpages, and click Add.

2. Select our structure, 1 or 2 columns.

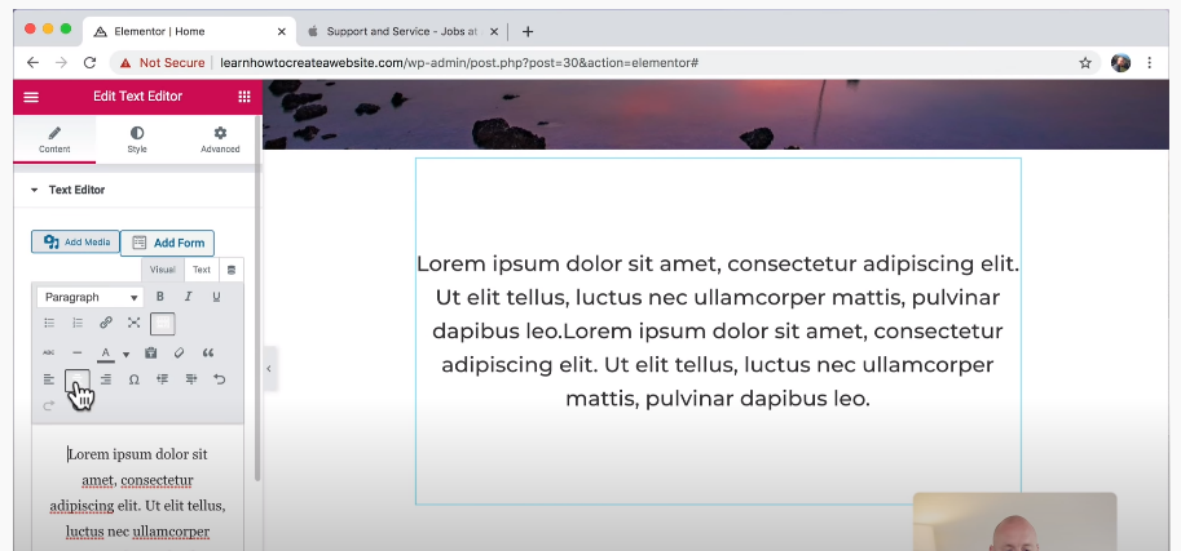
3. Choose some element that you want and drag it. You may customize the content of the element.



4. We may add an image below our text. Add widgets. Choose height into minimum height. We may upload a file as the background. Change the size into cover so the image will not be cut.

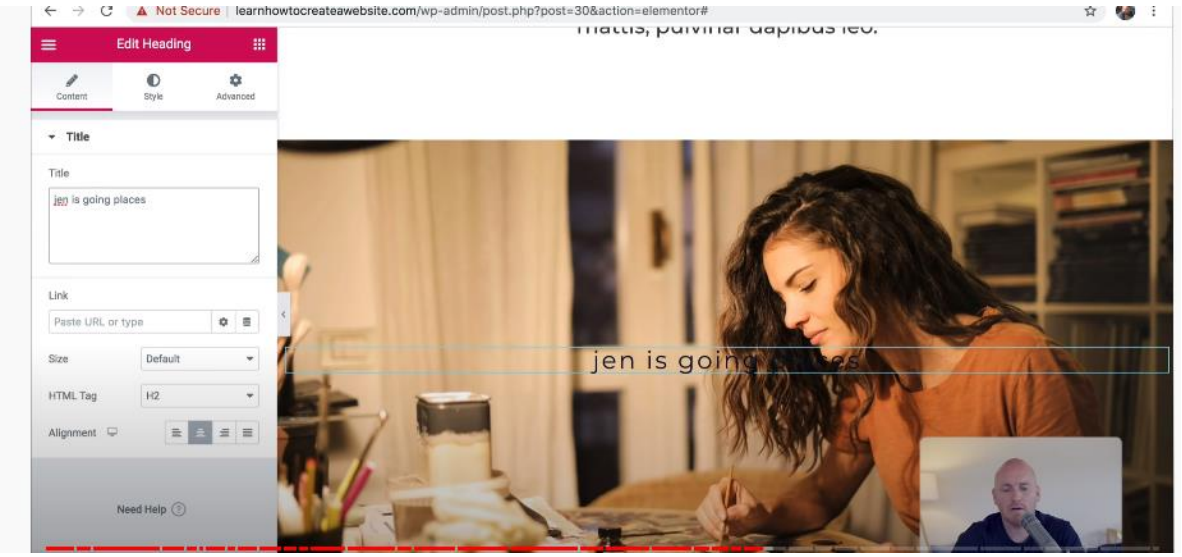


5. Add text below the image, just add widgets, and choose text element. We may customize the element like below.



6. We can duplicate the element before (such as an image element), right click, duplicate.

7. Change the image by clicking the Styles and choose the image, click Save and Insert. After that, we may also add some text by dragging Heading element and write the text.



8. Furthermore, we also may add button, put a link on the button (Content menu). Click add icon button to add a new icon.

9. We also can add a video by clicking video button and put the youtube link. (May change the start and end time)

Step 21 - Design Like Disney

Step 22 - Don't Design At All

Step 23 - Contact

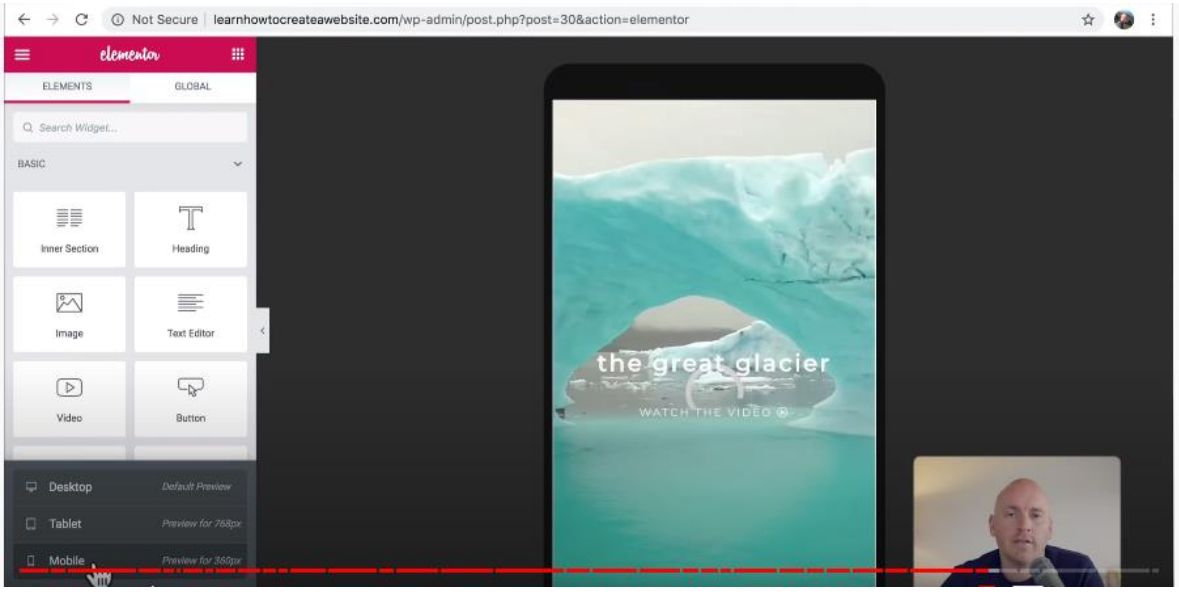
1. Change your email, map
2. If we want to add a new form, Go to WPForms, Add New
3. We may modify the contact element.
4. Modify notification for the email

- Fill the send to email address with your email.
- From name, fill it with name (show smart tags)

Section 6 - Making It Perfect

Step 24 - Mobile Friendly

1. Go icon desktop at the bottom part, choose the elements into Mobile.



2. Click on the video, choose the Elementor, style, and click on Play On Mobile, change the icon into on.

3. We also may change the arrangement of the picture and text, go to Advanced, Responsive, and make the Reverse Columns (Mobile) into on.

4. Click Update.

5. Change the button color on the phone. Click customize, header, transparent header, go to menu, choose Hover, choose Link Active / Hover Color into black / grey.

Step 25 - Footer

1. Go to Footer, click Edit Elementor, Site Footer.

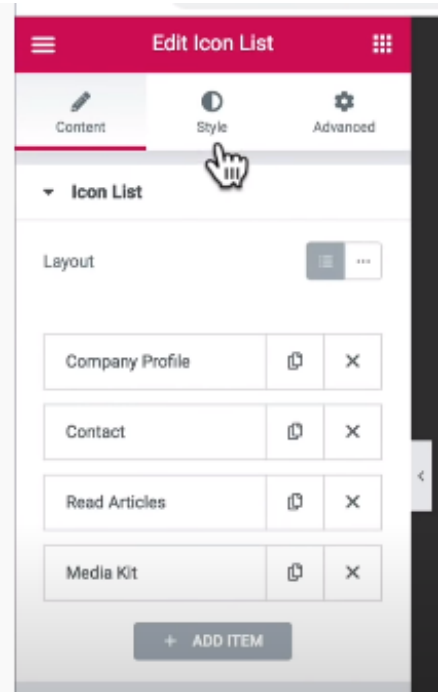
2. Modify the pages like the layout you want.

3. Go to Mobile setting, modify the link (navigation) (left to right, or up to bottom)

4. Modify the navigation first, one for desktop, and one for mobile.

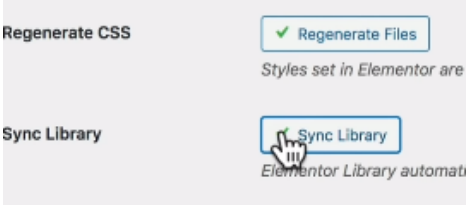
5. First, setting the first link Hide On Mobile (left to right)

6. Second, setting the second link Hide On Desktop and Hide On Tablet. (Advanced, Responsive). Go to the content, change it into List Layout (up to bottom). You also may change the paragraph.



Step 26 - SSL

1. To make our website secure, so it makes sure all the user data will used only for the website owner, such as credit card information.
2. Go to Dashboard, Plugins, Add New. Type SSL.
3. Choose Really Simple SSL, Install Now, Activate
4. Your website just get a certificate, but is not enabled yet, you need to contact the hostinger.
5. When we go to our website, the http will change into https but still not secure because some images that we upload is using HTTP link instead of HTTPS link.
6. Go to the Elementor, Tools, Regenerate CSS and Sync Library



7. Replace URL: Copy our new URL (with HTTPs) and paste it in Update Site Address (URL) (Right) and paste our old URL (with HTTP) and paste it in the left column. Click Replace URL, it will show how many rows affected by changing from HTTP into HTTPs.

Day 1 - Tuesday, 1st December 2020
Understanding Themes And Plugins

1. WP Ocean

https://oceanwp.org/demos/

OceanWP is the perfect theme for your project. Lightweight and highly extendable, it will enable you to create almost any type of website such a blog, portfolio, business website and WooCommerce storefront with a beautiful & professional design. Very fast, responsive, RTL & translation ready, best SEO practices, unique WooCommerce features to increase conversion and much more. You can even edit the settings on tablet & mobile, so your site looks good on every device. Work with the most popular page builders as Elementor, Beaver Builder, Brizy, Visual Composer, Divi, SiteOrigin, etc... Developers will love his extensible codebase making it a joy to customize and extend. Best friend of Elementor & WooCommerce.

Step

1. Install Ocean WP and Elementor
2. Run the Setup Wizards, choose the demo template site, then install it. Import XML Data.
3. Customize, we may change the Site title and tagline, etc. Continue.
4. Ready, will produce clean design.

2. PostSMTP

https://wordpress.org/plugins/post-smtp/

Post SMTP is a next-generation WP Mail SMTP plugin, that assists in the delivery of email generated by your WordPress site. Post SMTP is the first and only plugin to support the latest security standards. With OAuth 2.0, there is no need to store your email passsword in the WordPress database where it might be found.

Step

1. Install WP Mail SMTP by WPForms, activate, and install.
2. Go to Settings, WP Mail SMTP
3. Fill your email on From Email, your blog in From Name, Mailer : choose other SMTP
4. Create our own email with @ourwebsite-domain.com
5. Fill SMTP Host with ourwebsite-domain.com
6. Encryption : SSL
7. SMTP Port : 465
8. Auto TLS On
9. Authentication On
10. On Email Test tab, try to click Send Email (Success or not)

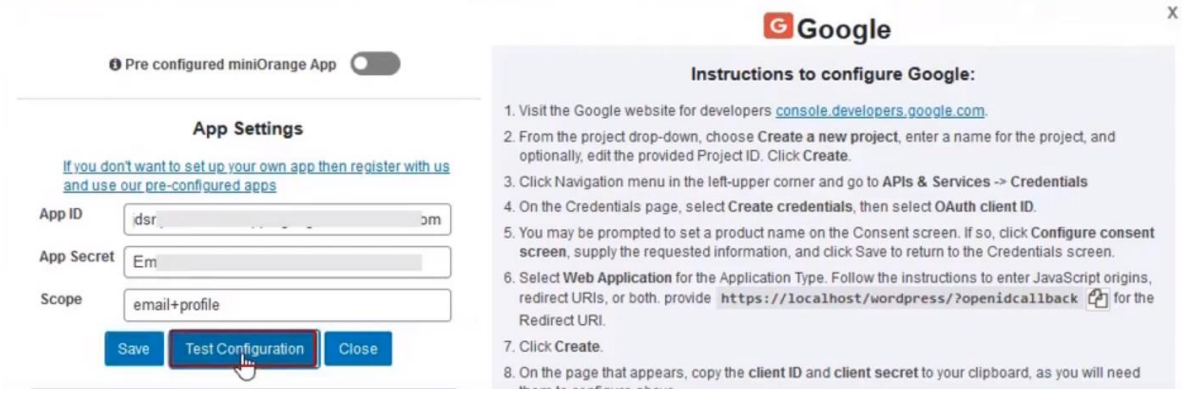
3. Social Login

https://tw.wordpress.org/plugins/miniorange-login-openid/

Social Plugin enables social login, social sharing, social comments using social login apps like Google, Facebook, Twitter, Vkontakte, LinkedIn, Windows Live, Instagram, Amazon, Salesforce, Yahoo, WordPress, apple, Paypal, disqus, pinterest, spotify, reddit, tumblr, twitch, vimeo, kakao, discord, dribbble, flickr, line, meetup, stackexchange, livejournal, snapchat, foursquare, teamsnap, naver, odnoklassniki.

Step: such as MiniOrange

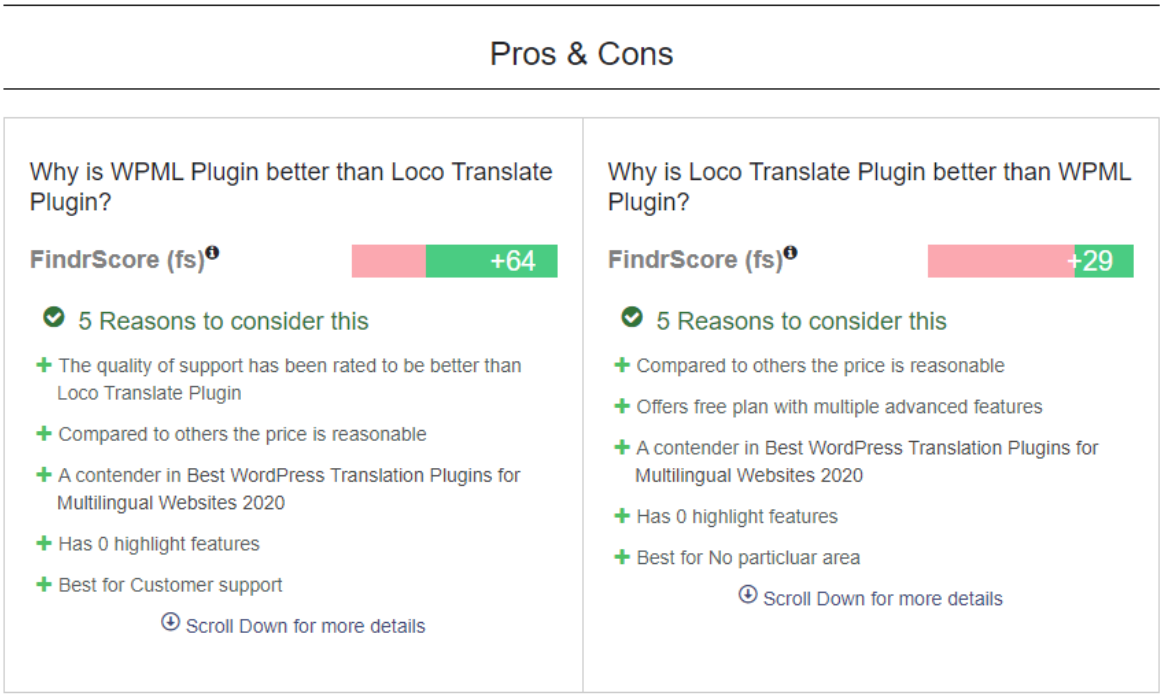
1. In plugin, click on icon you want to connect.
2. Register with miniOrange or just skip it
3. Click on the link on number 1



4. Click Test Configuration to test it.

5. But if you can log in using your account, you may enable other social media without make a setting of each icon.

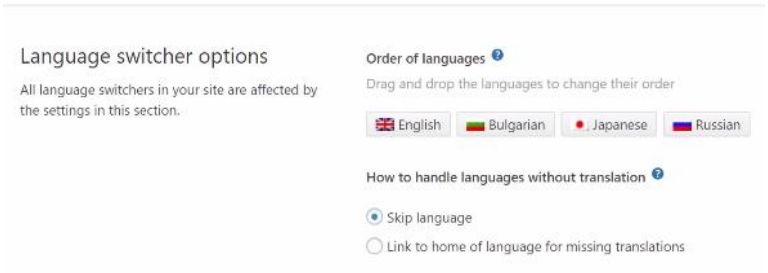
4. Loco Translate / WPML



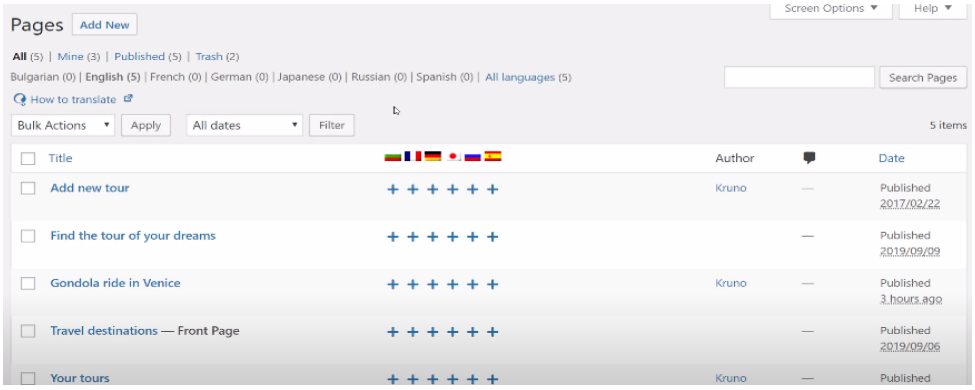
WPML Steps:

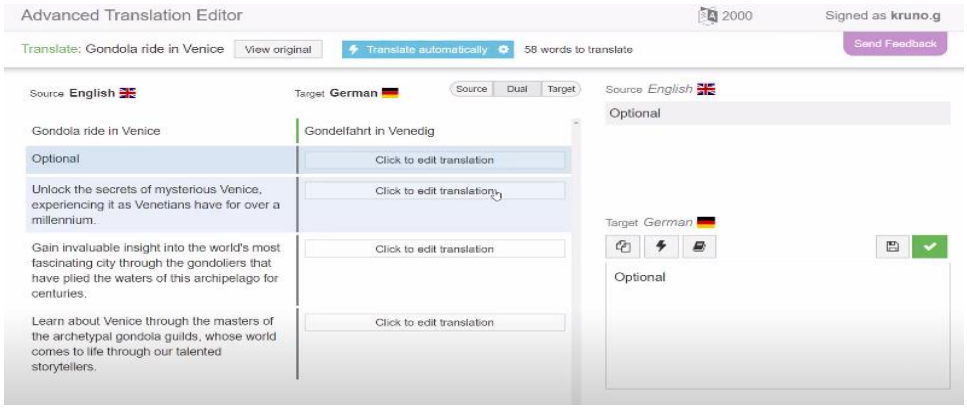
1. Install Plugin and Setup it.

2. Choose some languages.

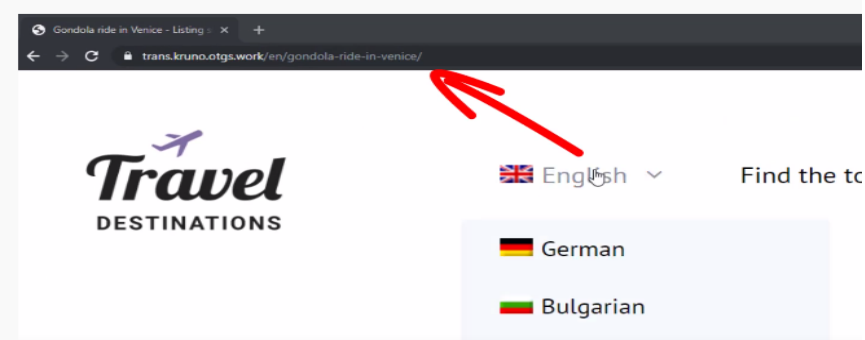


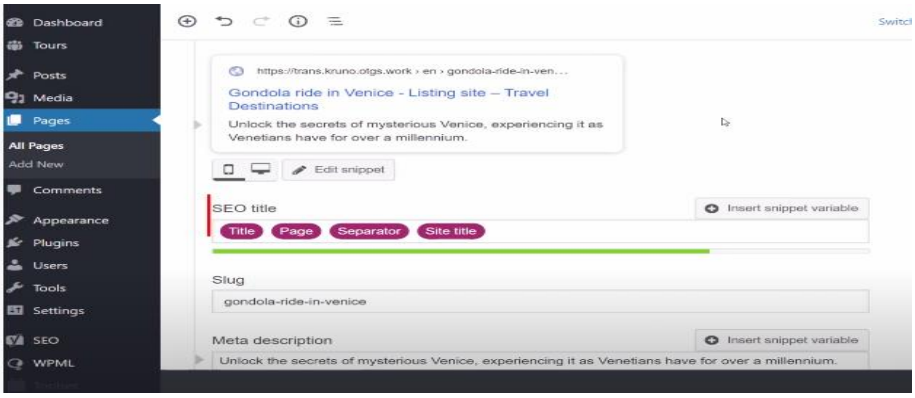
3. Content Translation



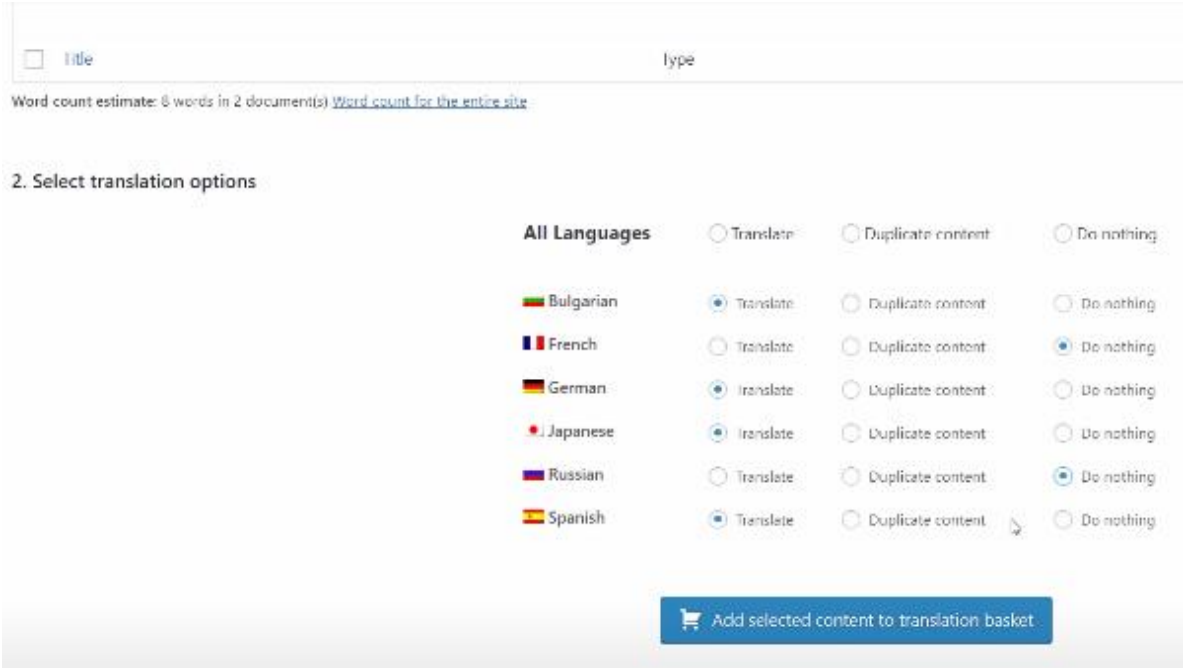


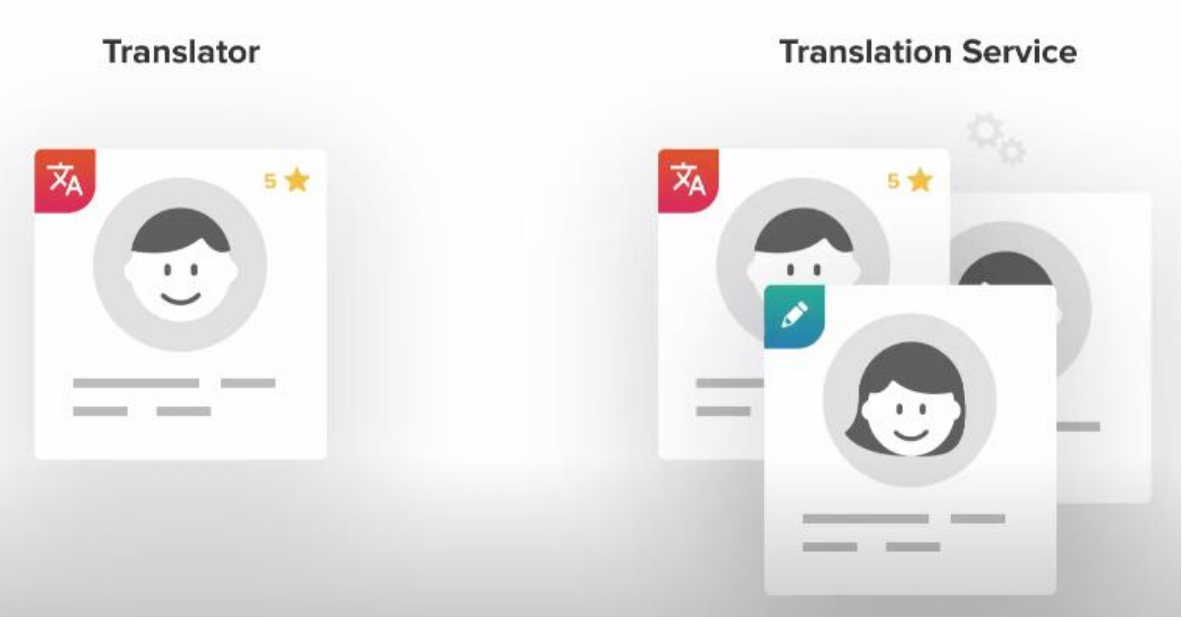
4. SEO Friendly





5. Setup Options





5. Woocommerce

https://www.businessbloomer.com/woocommerce-visual-hook-guide-single-product-page/

https://woocommerce.github.io/code-reference/index.html

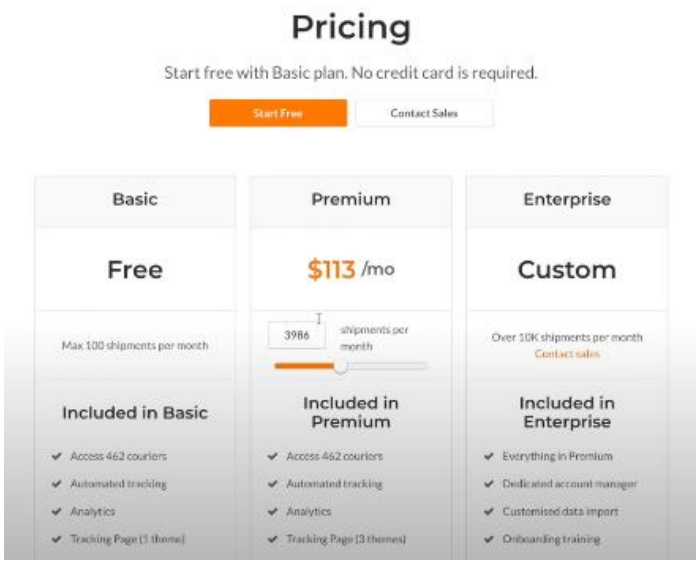
In Wordpress, the pages can be divided into two styles. First, in one page, Second, in some pages.

Facilities:

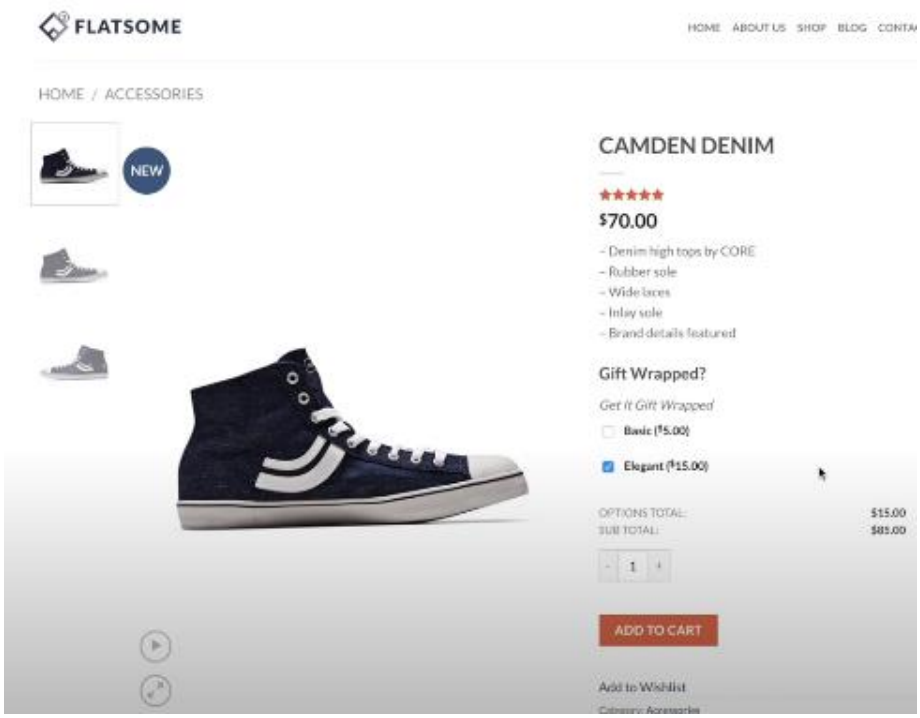
1. After Ships

To input tracking information on products so our customers can actually track the products instead of them contacting us. Click 'Start Free'.

We can modify the package and pricing like below.

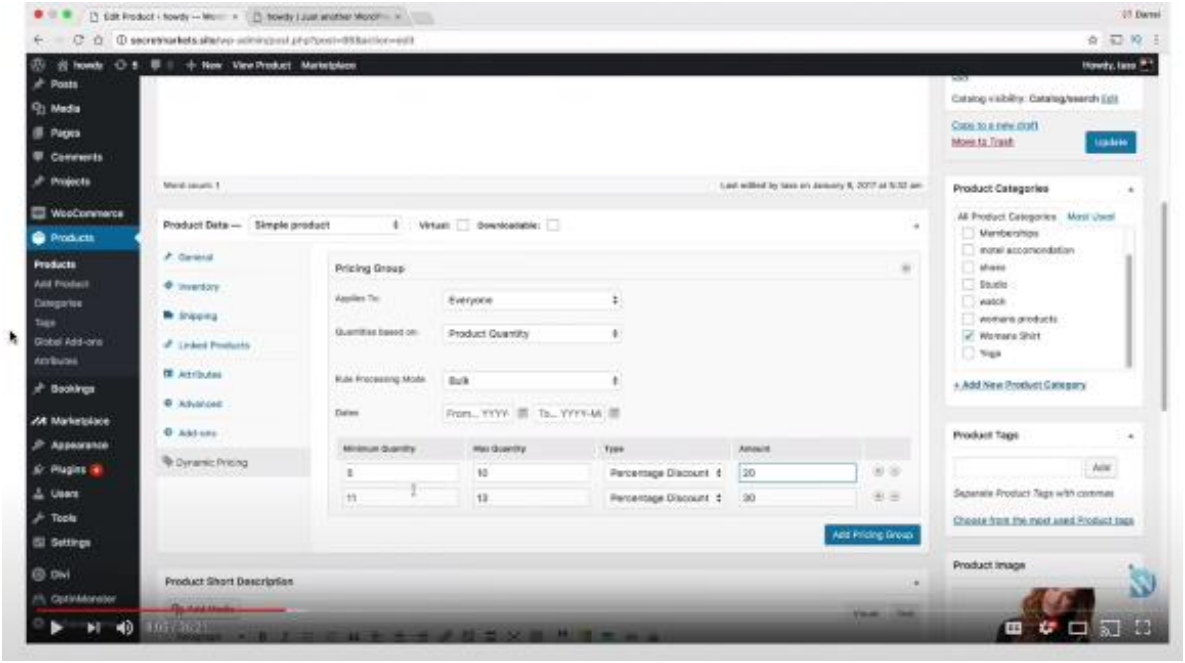


2. WooCommerce Product Addons



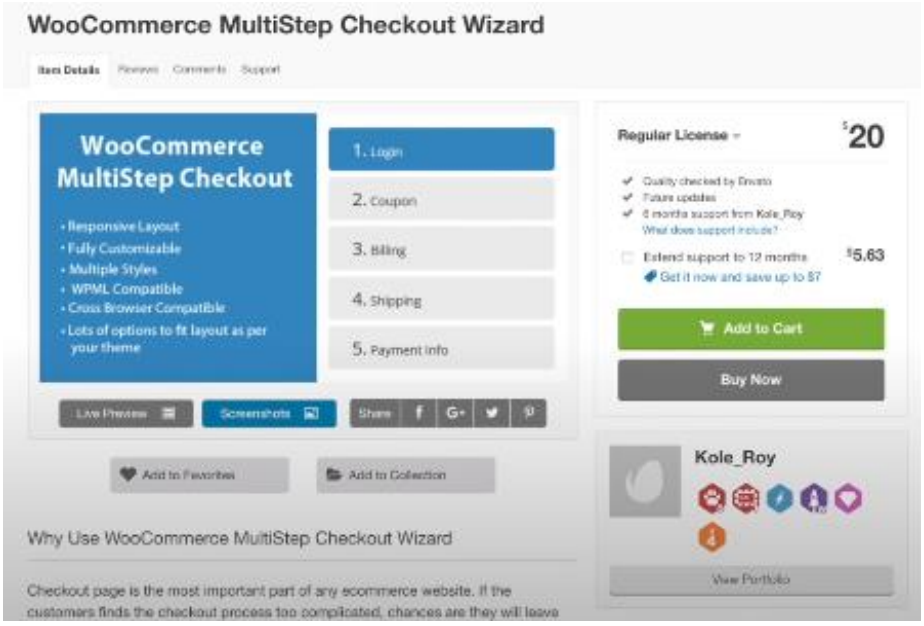
3. Dynamic Pricing Plugin

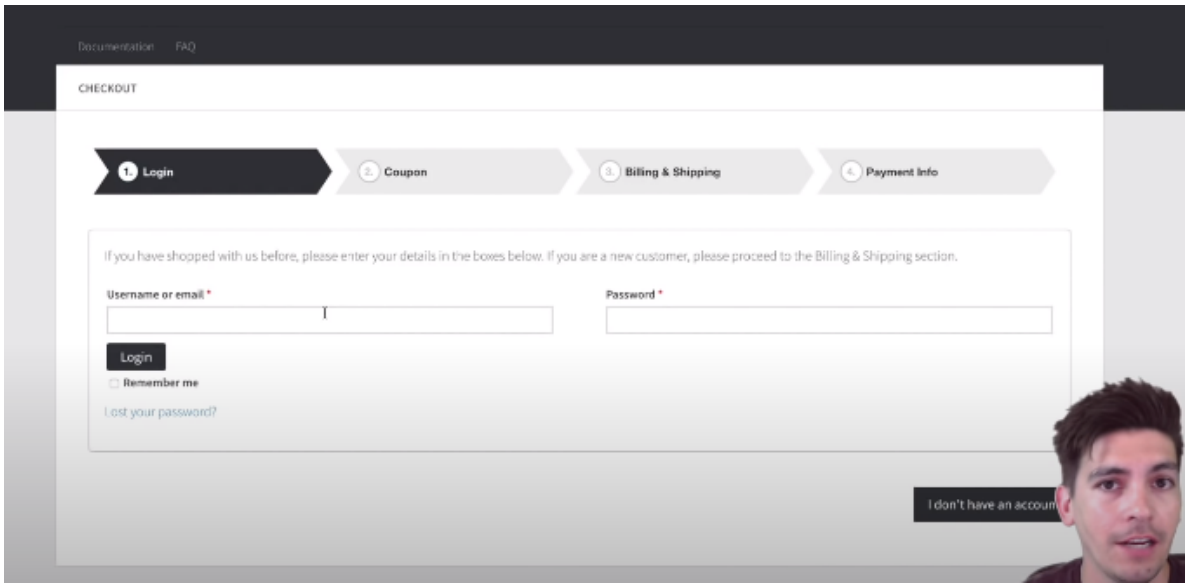
Customize buy 1 get 1, etc



4. Multi Step Checkout

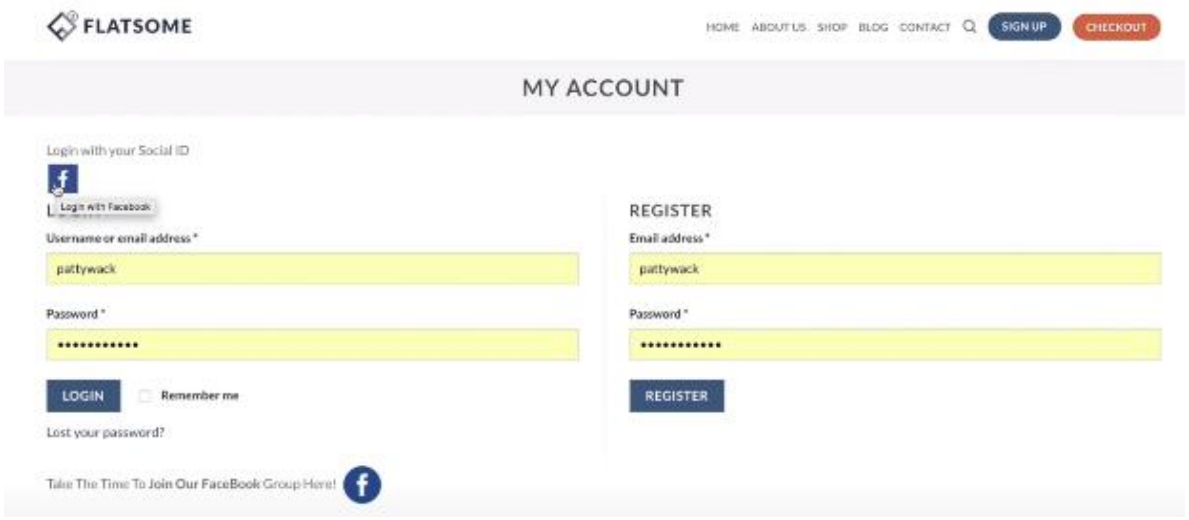
They provide multi step and make the payment secure.





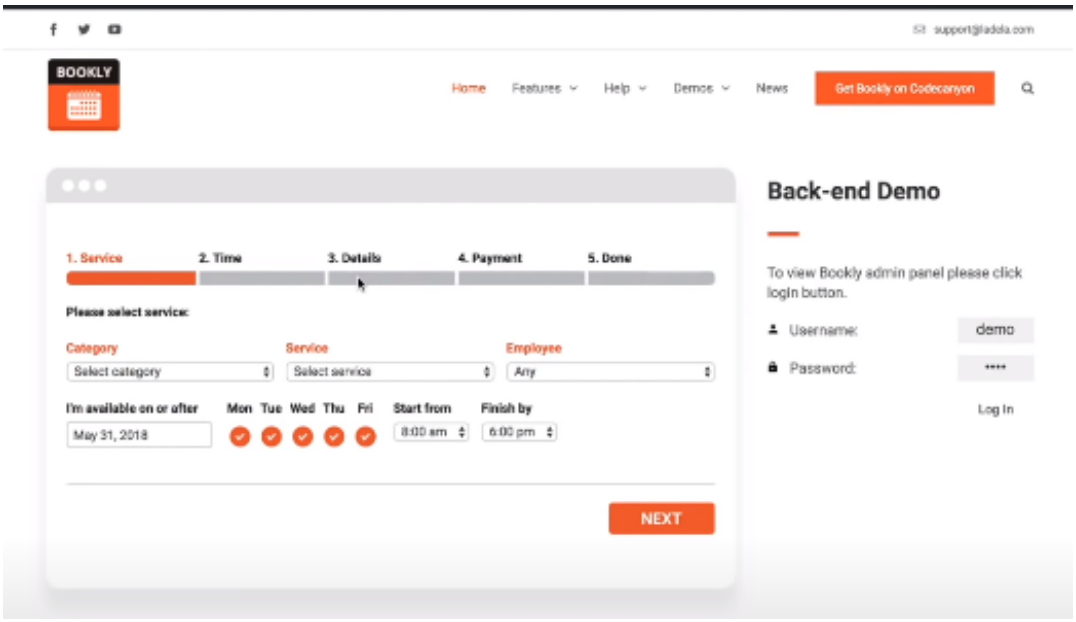
5. Super Socializer Plugin

We may login for any social media.



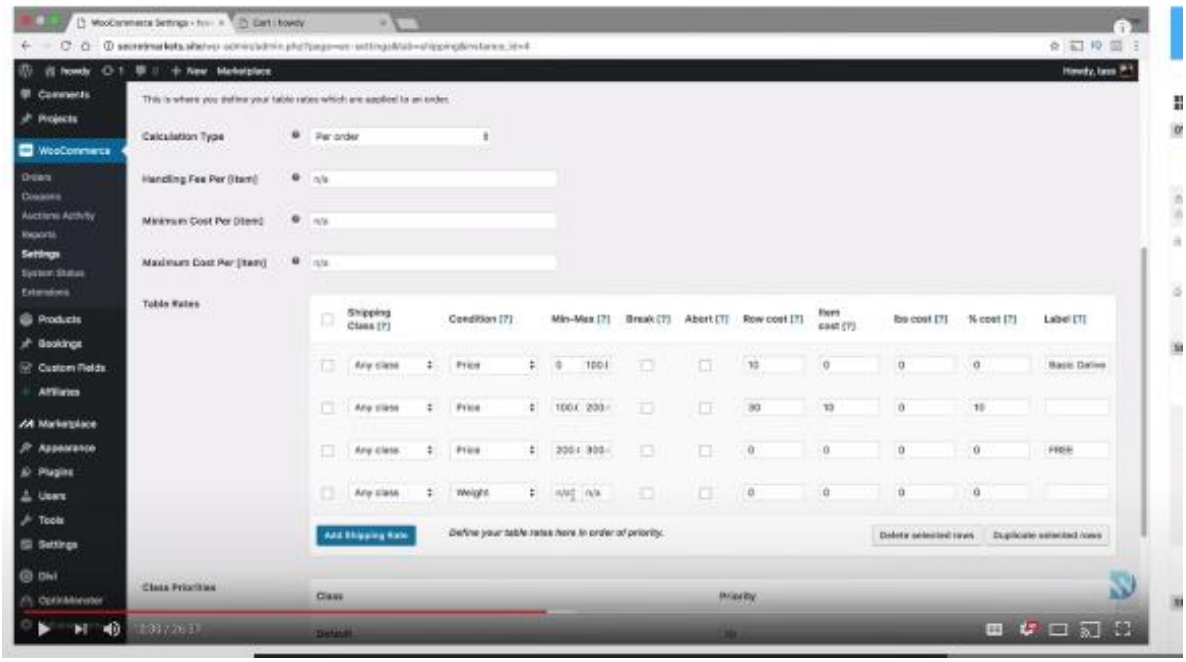
6. Email Customizer (?)

7. Bookly Plugin



8. Table Rate Shipping

We may modify the shipping price based on what we think.

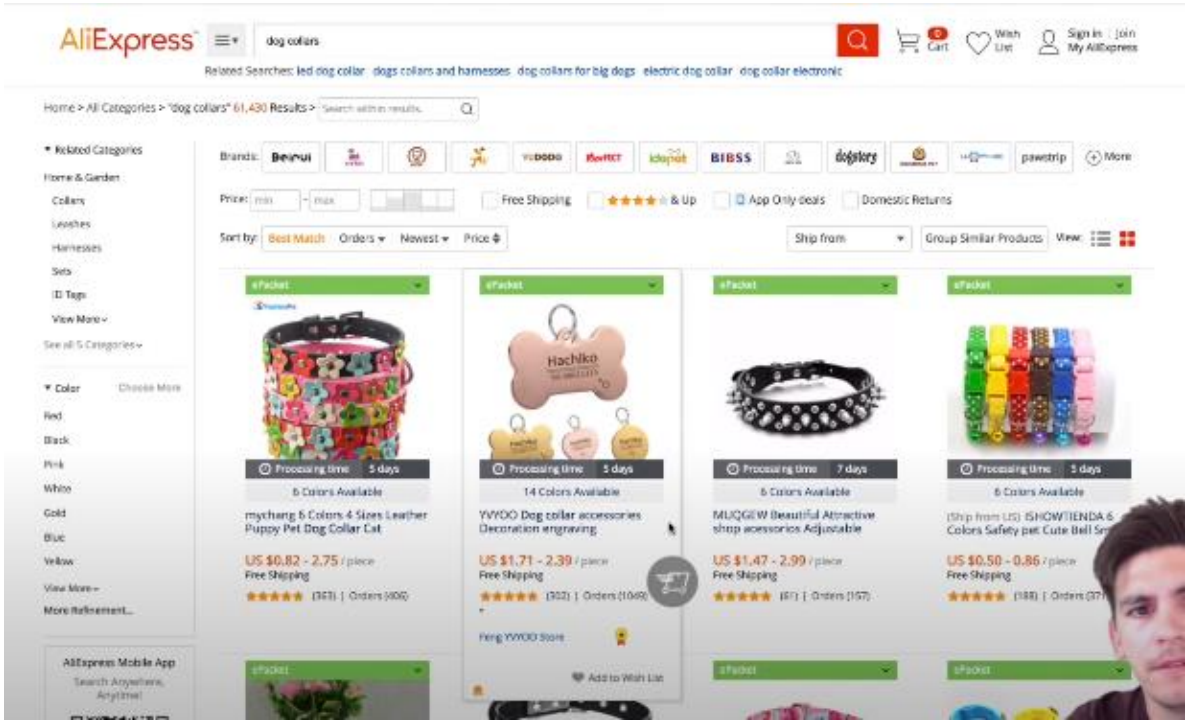


9. WooCommerce Facebook

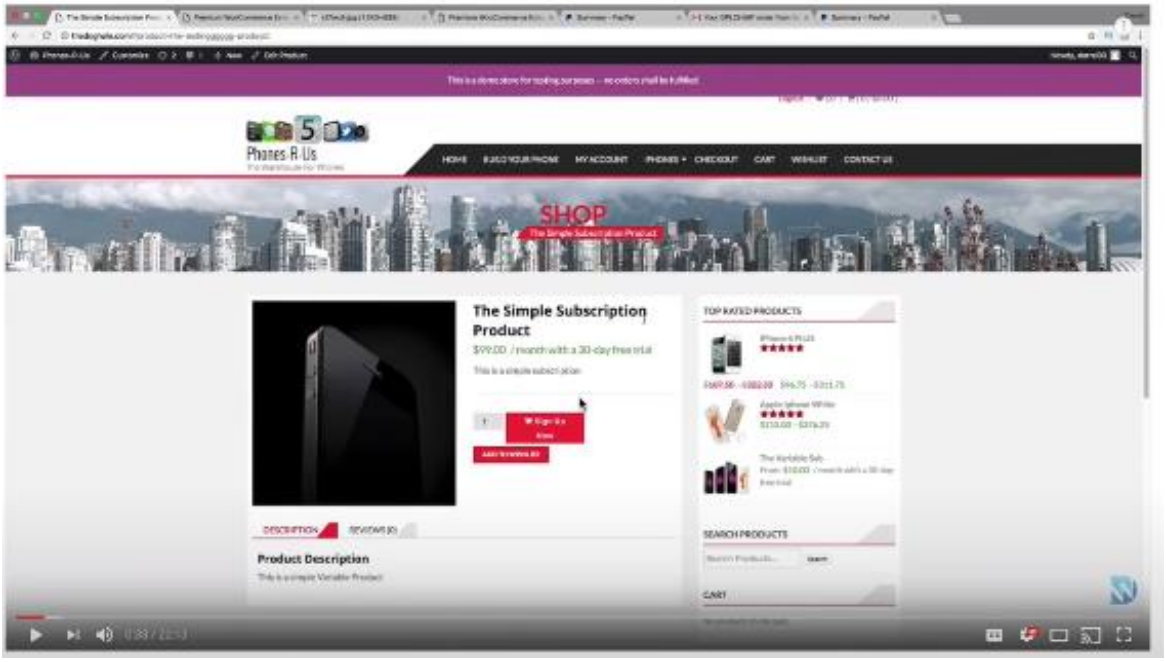
Connect to facebook

10. Dropshipping Website

As a dropshiper we may use this plugin. We also may make some settings for the dropshipping rules.

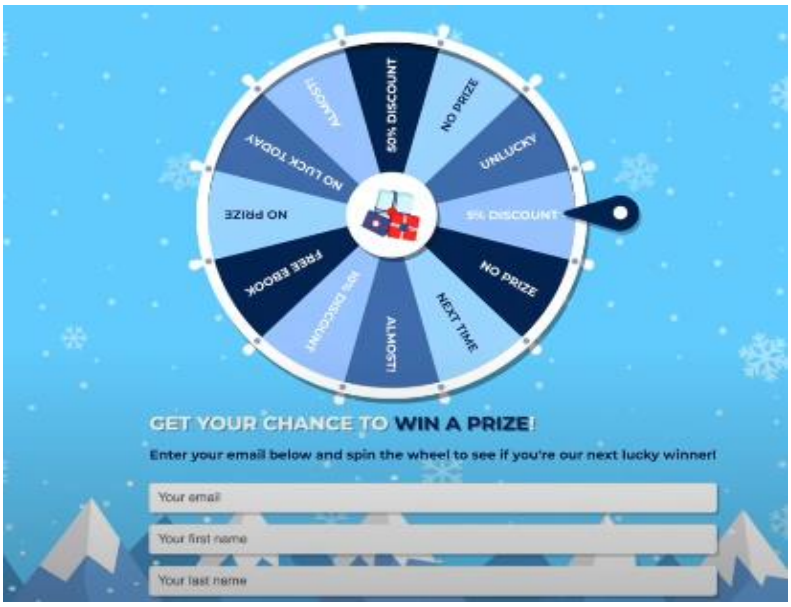


11. WooCommerce Subscription (Membership)



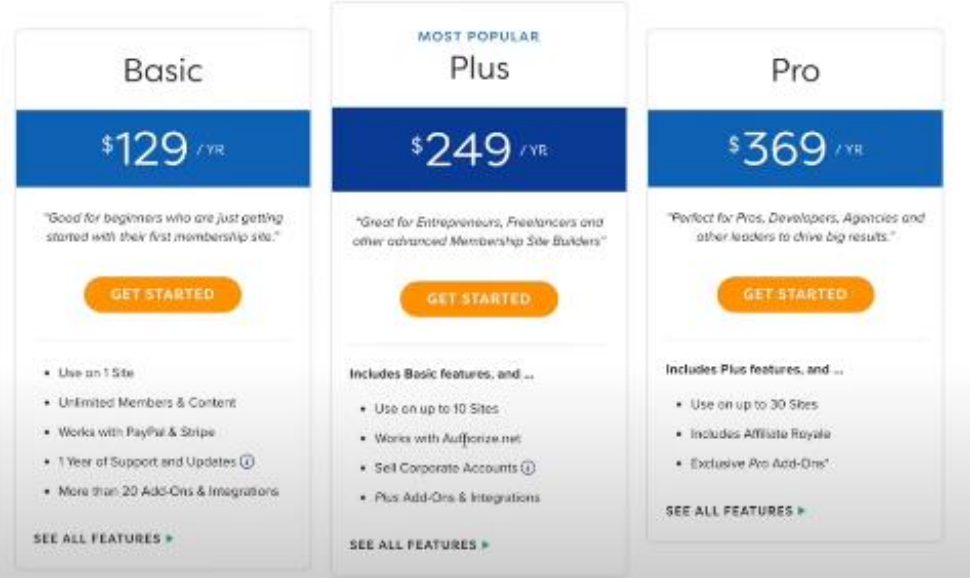
12. MailChimp for WooCommerce

13. Wheels for Lottery (Get discount, etc)



14. MemberPress Plugin

Give some membership package, we may modify this.



6. Woocommerce YITH Plugins https://yithemes.com/

- Dynamic Pricing and Discounts Premium
- Points and Rewards Premium
- Wishlist Premium
- Product Bundles

7. Elementor Pro

Day 2 - Wednesday, 2nd December 2020
Elementor Game Website

依照範例網站製作三個頁面:

範例網站 : https://www.pgsoft.com/cn

說明: 遊戲使用”文章”進行製作即可

- 遊戲列表 2. 遊戲時間軸 3. 遊戲篩選 (皆帶有 Banner Slider 功能)

- 遊戲列表:

  顯示所有遊戲列表，依照分類顯示 文章(遊戲)slider

  範例網站的連結 : https://www.pgsoft.com/cn/games/home/

- 遊戲時間軸: 使用外掛(Infinite Timeline)

  依照時間顯示遊戲的順序，由最新排到最舊

  並且可以依照年度顯示該年度所有遊戲

  這個功能需要修改外掛本身建立 shortcode 的內容與樣式

  (外掛目前遊戲是採用左右左右排序 需要像範例網站一樣 圖片擺在右側)

  在外掛功能中增加 可以使用”年份”參數來決定顯示該年份的文章

  並且頁面左側會有時間線的 fliter

  點擊該年份，即可動態更換頁面的遊戲內容

  範例網站的連結 : https://www.pgsoft.com/cn/games/timeline/

- 遊戲篩選: 先不需要製作右上角的顯示方式更改 與 順序更改功能

  一開始是所有遊戲顯示列表 (與第一點的樣式不一樣)

  先不需要做當文章 hover 時的動態效果 單純顯示圖片 與標題即可

  左側篩選是依照文章分類去製作 分成大小分類 篩選

範例網站的連結 : https://www.pgsoft.com/cn/games/all/

Create the pages below based on the example given:

Example: https://www.pgsoft.com/cn

Reminder: Use “post” to create game item.

Pages:

1. Game list
2. Game Timeline
3. Game Filter

*Each page is required to include Banner Slider

1. Game List:

a) Show all the game list, show the game item(post) sorted by categories

b) Example: https://www.pgsoft.com/cn/games/home/

2. Game Timeline: (Use plugin *Infinite Timeline*)

a) Show the game items ordered by time, from the most recent item to the oldest.

b) Show the game items based on year.

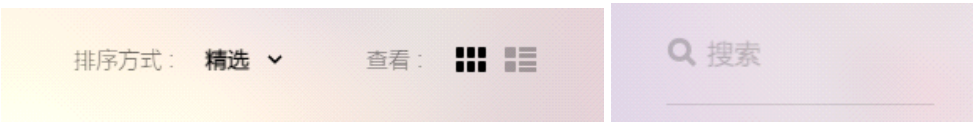
**This feature required modification of plugin shortcode in term of its function and style. The default setting of plugin arranges the items by left and right. However, you are required to arrange all the item picture at left, just like the website given(refer to the link given below).*

c) Add a new feature that enable user to decide the display of the game items(posts) based on the year selected. And the year selection filter shall be placed at the sidebar on left.

d) The expected outcome is to switch the display of game items when a particular year is selected without having to go to another url.

Example of timeline display: https://www.pgsoft.com/cn/games/timeline/

3. Game filter



a) Sorting method, appearance mode selection and search bar above are not required.

b) By default, this page shows all the game items. But the appearance style is different from Page 1 Game List. (Refer both links for comparison).

c) Like the example below given, show the picture and title of each item. The hover effect for each item is not required.

d) Please use post categories(parent category and child category) to create the filter sidebar on left in the example website.

Example for game filter: https://www.pgsoft.com/cn/games/all/

Learning Frontend Process

1. Preparation

1. In the beginning of every project, you MUST decide which type of developer website will you use:

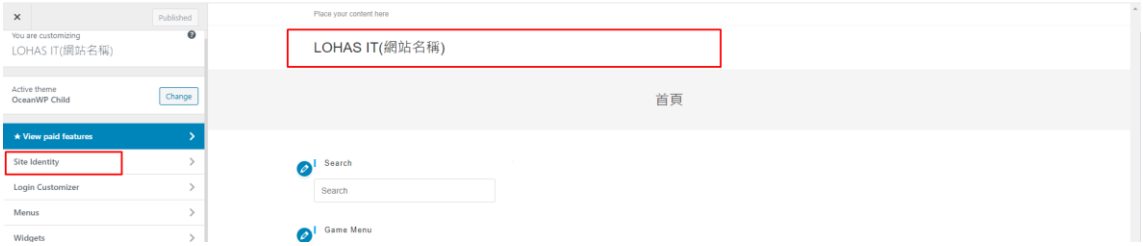
    1. alfa1 : 1st Theme : (?)
    2. alfa2 : 2nd Theme : OceanWP
    3. beta1 : 1st Theme : (?) with shopping cart
    4. beta2 : 2nd Theme : Ocean WP with shopping cart

2. After you decide, you need to decide which part (layout) you need to make using the theme (OceanWP) or Elementor. In this example, the slider (upper pages) and sidebar, you will use the theme template because it will appear in every pages. Elementor just use to create a content. In this example, we will use alfa2 because we dont need shopping cart, but we also can use beta2 and hide the shopping cart plugins.

2. Create Slider

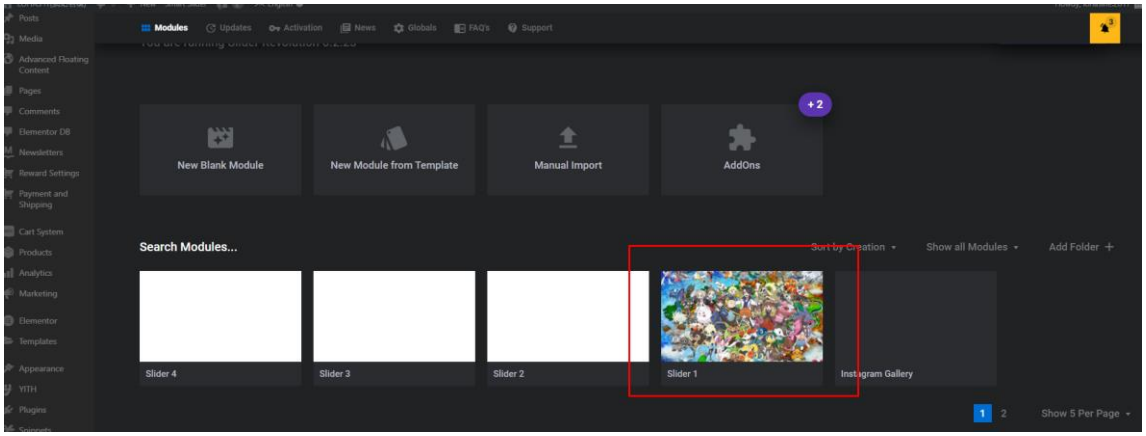
1. From the web, you click Customize.

2. Site Identity is used to change the Title and Tagline.

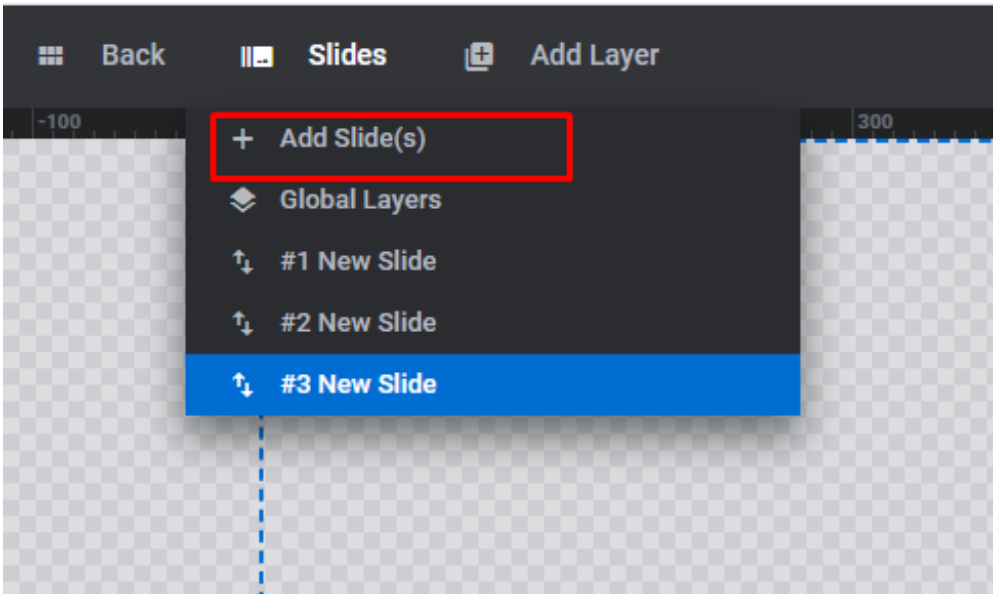


3. Click Dasboard, RevSlider. Create a new slider (or Edit). (Global Layer because your slider will appear in every pages). Save the shortcode to put it in the layout later.

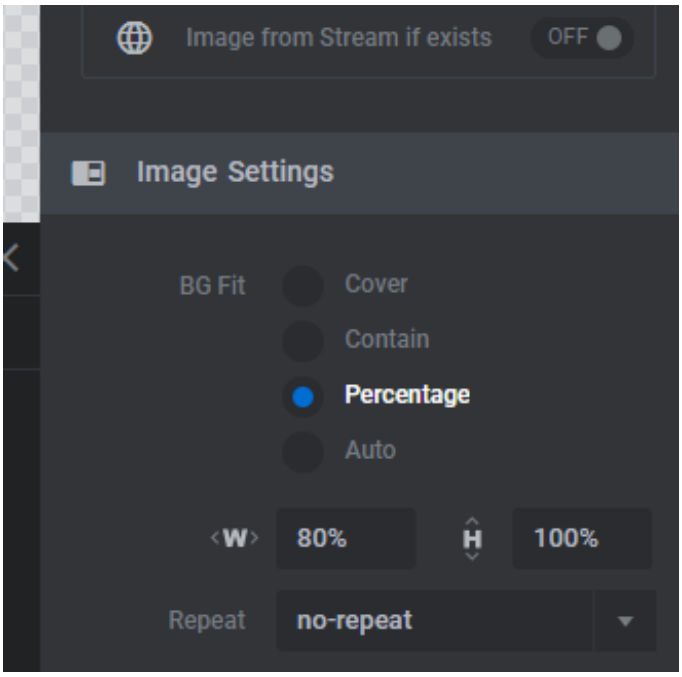
[rev_slider alias="slider-1"][/rev_slider]

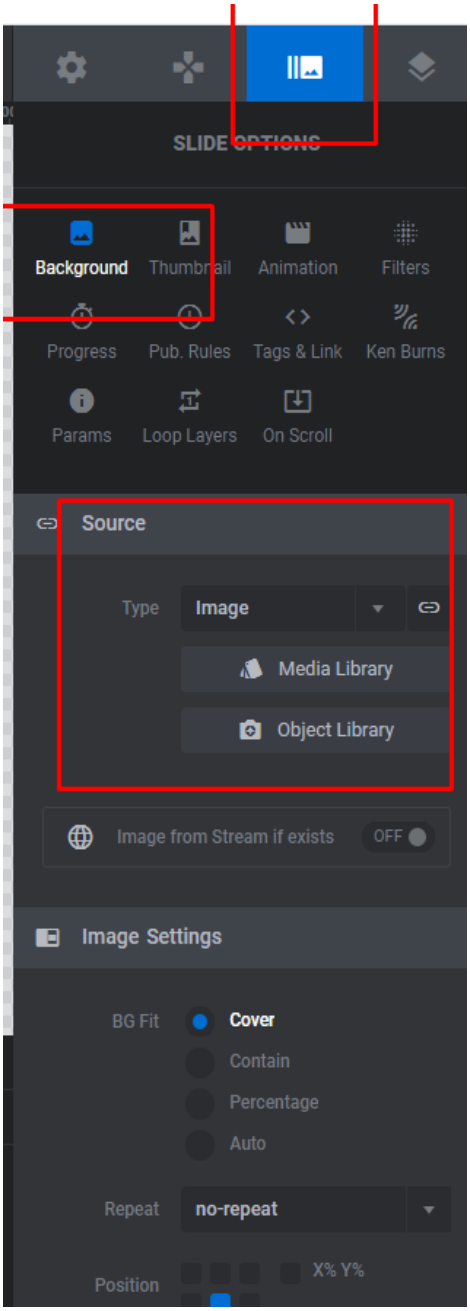


4. Create a new slide to add new slide and put the image there.



To make some size setting, you may use this.

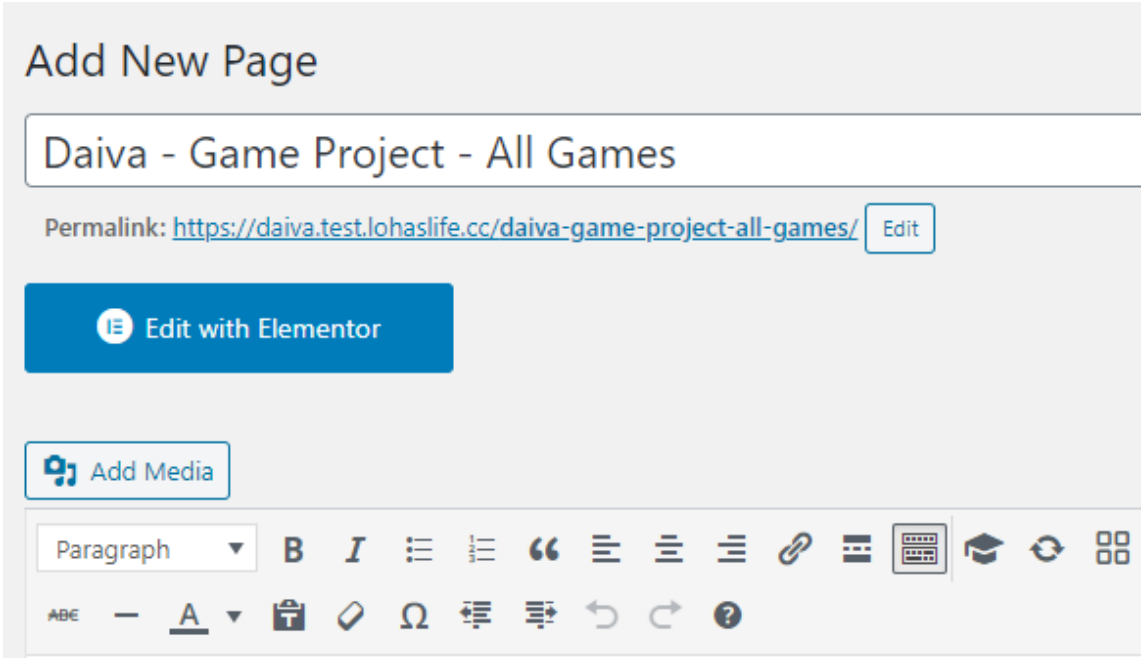


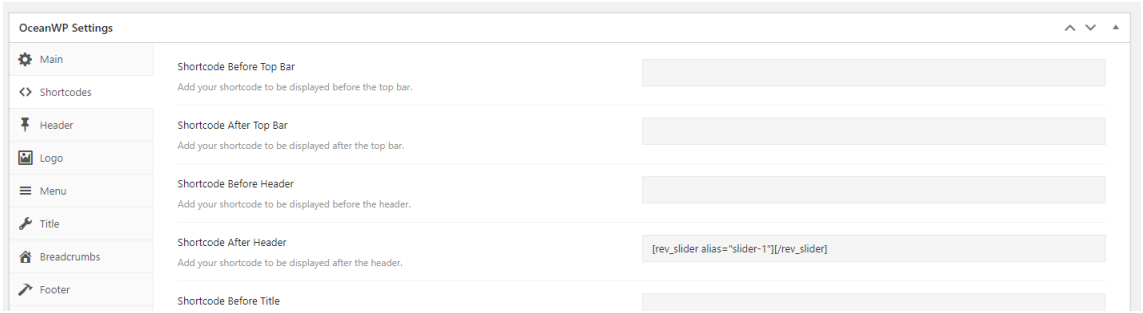


To add the background, you may use this.

Save it.

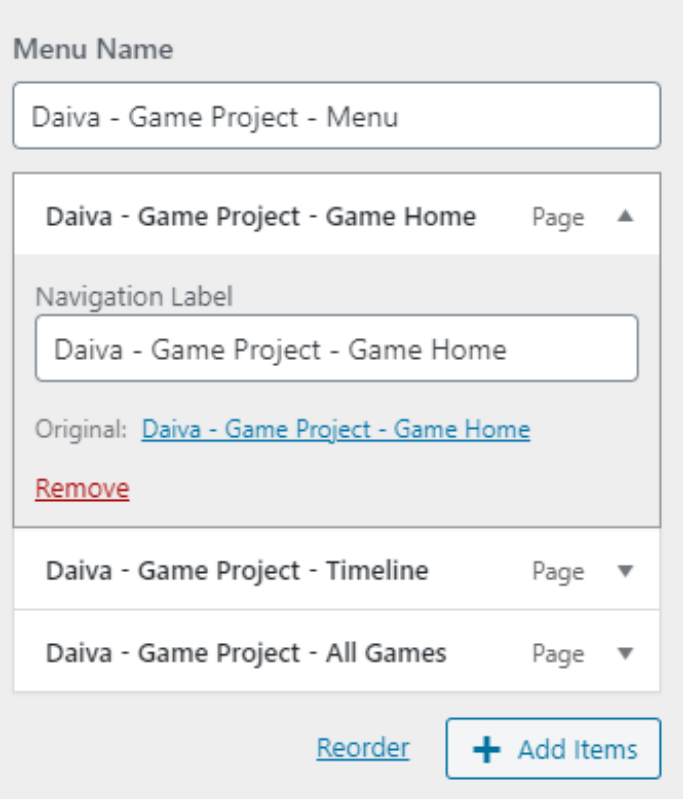
5. Create a new page (3 pages in this example: Game Home, Timeline, All Games) and put your shortcode of the slider. Publish.



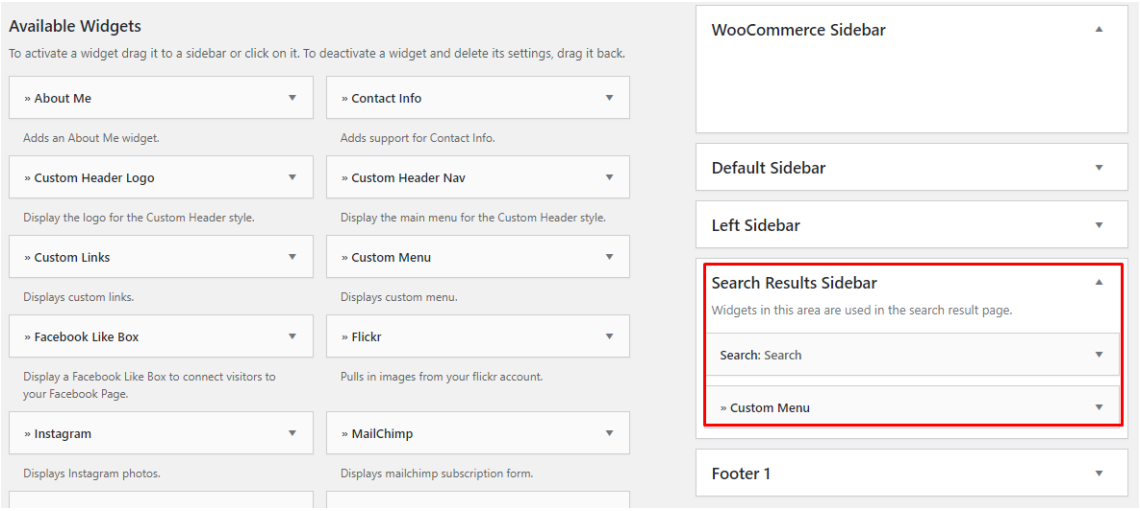


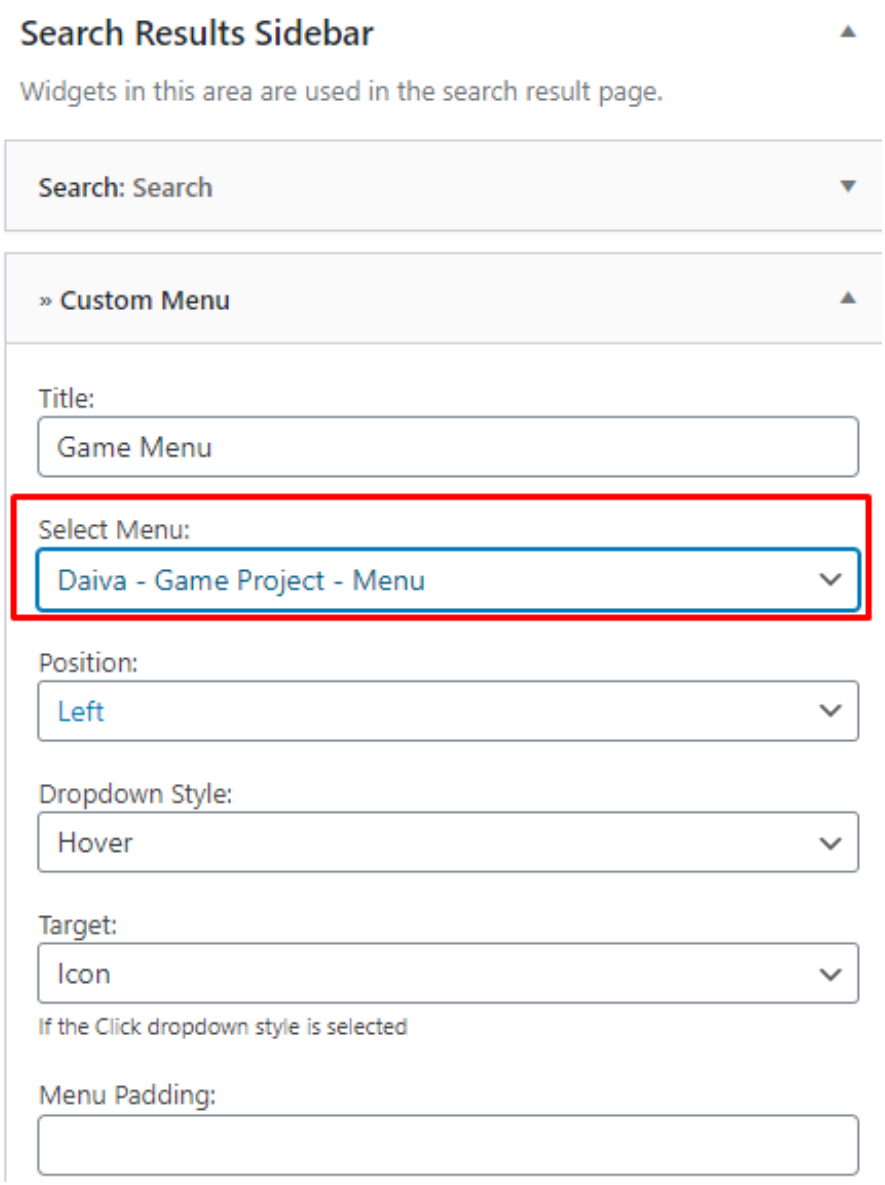
3. Create Side Bar

1. Click Customize, Menu, Create New Menu. Add All the pages that have you made. Publish.

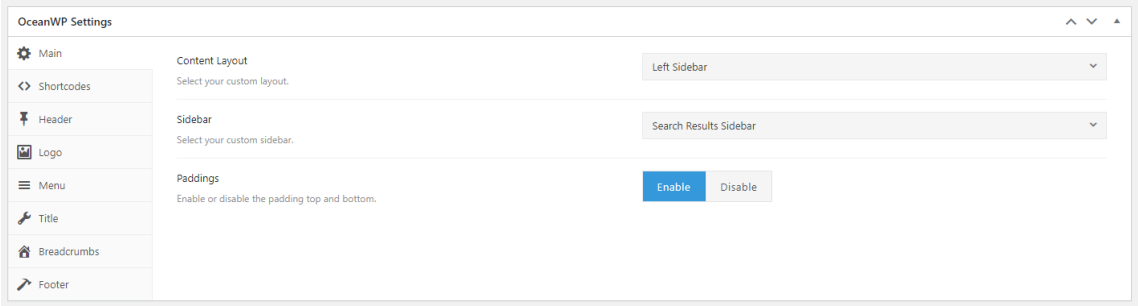


2. To put your menu into sidebar (on your website), go to Dashboard, Appearance, Widgets. Choose Search Results Sidebar and put Search and Custom Menu. (Drag) Save.

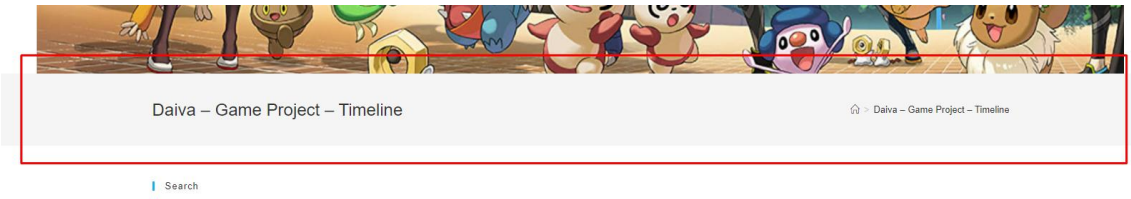


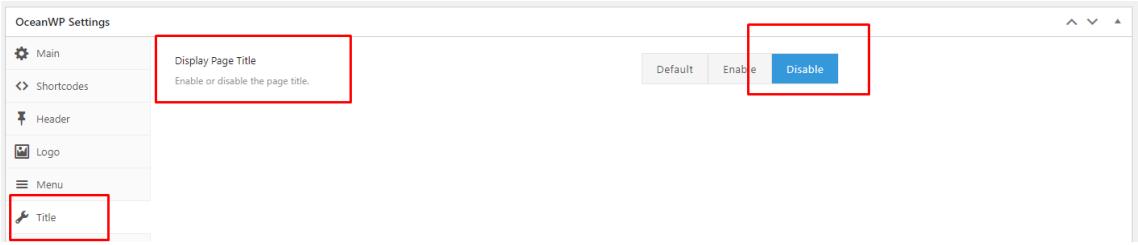


3. Go to Pages which you want to put the sidebar and make the setting like below. Update. Do this for all the pages.



4. To Delete this part, you need to make a setting below on every pages.





4. Make the content of every pages.

A. Game Home to show the categories of game.

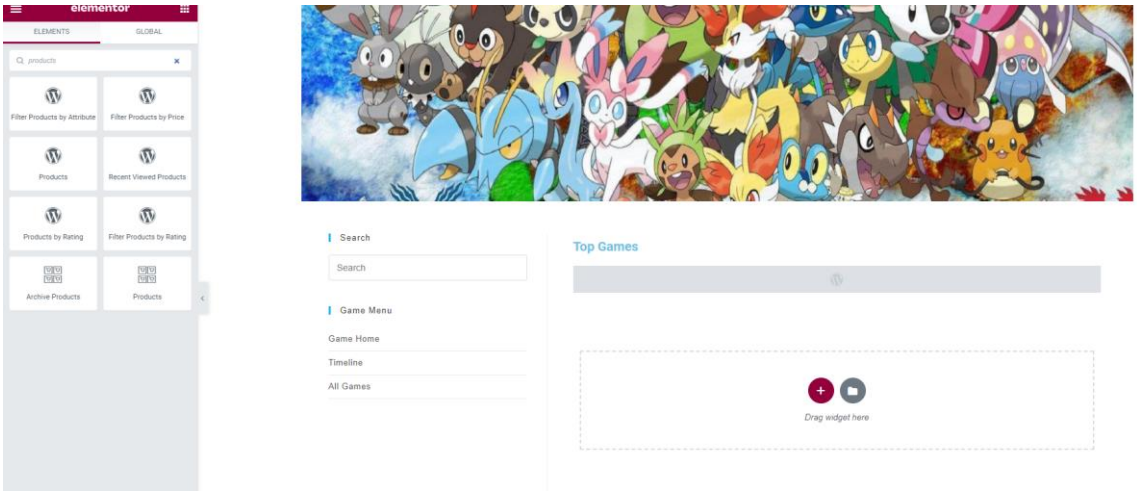
1. Go to the pages that you want to make and put some text header and products categories (archieved).

B. Timeline to show the time line.

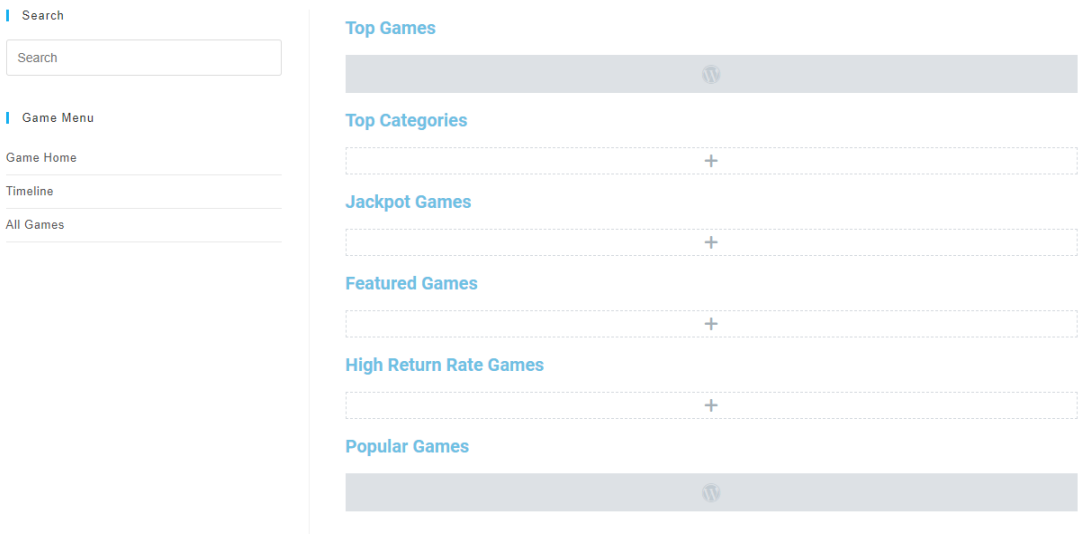
1. Install infinite plugins

2. In your pages, put shortcode on the elementor, and paste this code below.

[infinite timeline]



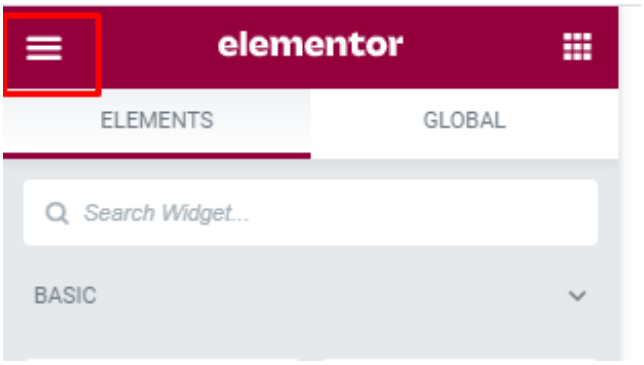
3. The result will be like below.

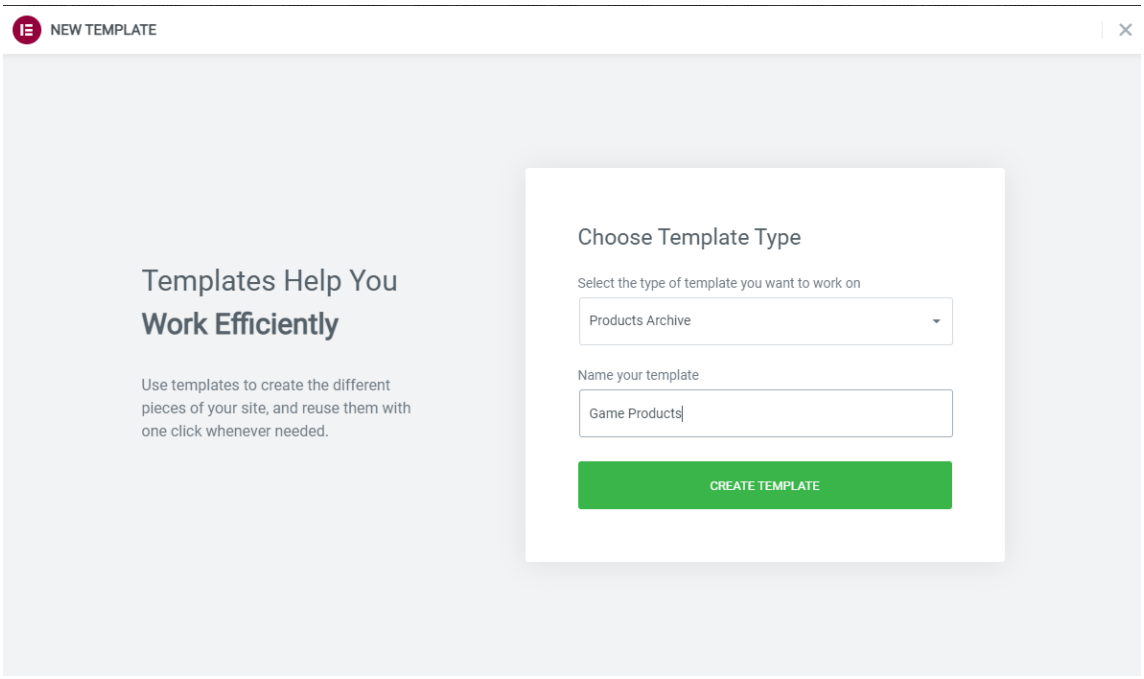


C. All Games To Show All Games.

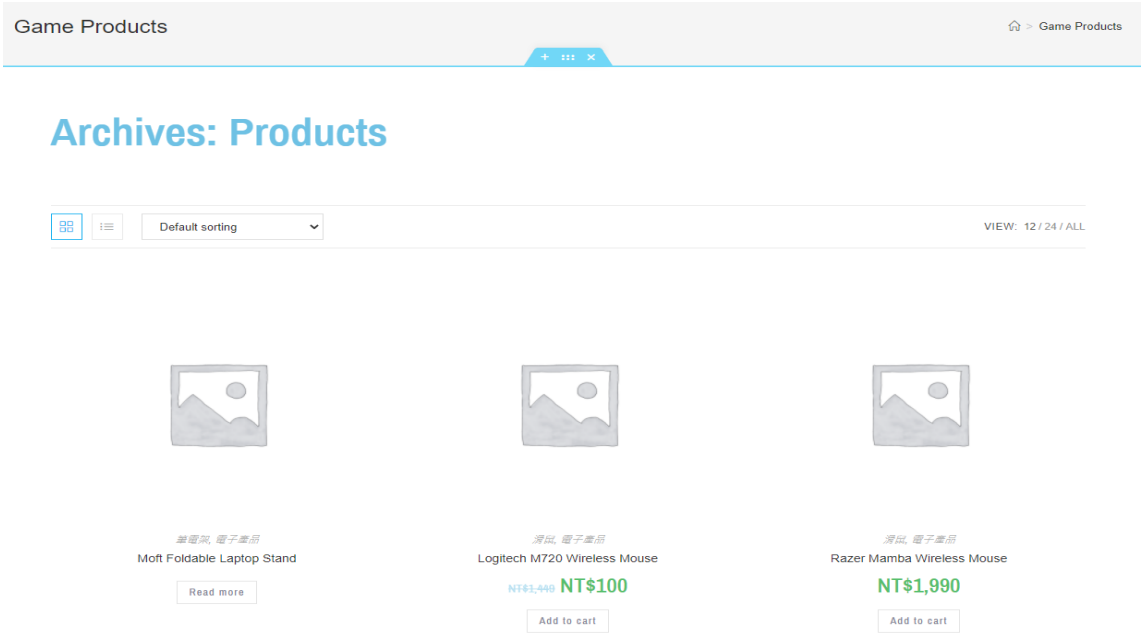
Because in this page you need to put all the game (products, not post) with the filter sort by, view, etc, so you need to create a template of the product first, instead of using Posts. (If you only use Posts, you just show all the post, and you may not use Sort by and View)

1. Go to the pages, Edit with Elementor. Click the burger sign, choose Theme Builder, Produtcs Archieve, Add New. or Dashboard, Templates, Add New. Create Template.

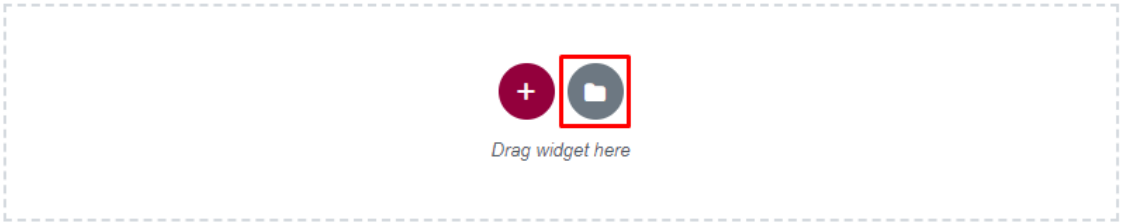




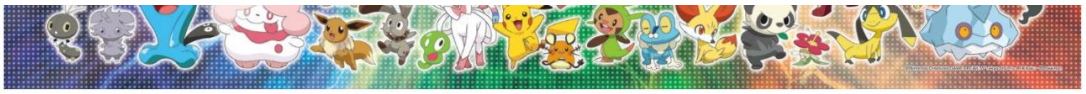
2. Choose the block that you want. (not from my templates). Publish.



3. Go to the Pages, Edit with Elementor, Choose the folder button and choose the template you want, Insert, Update



Game House



Search

Search

Game Menu

Game Home

Timeline

All Games

Top Games

Top Categories

Jackpot Games

Featured Games

High Return Rate Games

Popular Games

Timeline

Search

Search

Game Menu

Game Home

Timeline

All Games

2020

2020-12-02
Content Game 3 – 1

2020-12-02
Content Game 2 – 1

2020-12-02
Content Game 1 – 1

2020-12-02
Content Game 2 – 1

2020-12-02
Content Game 1 – 2

2020-11-26
Blog 1

All Games

Search

Search

Game Menu

Game Home

Timeline

All Games

QUICK VIEW

Moft Foldable Laptop Stand

Read more

Logitech M720 Wireless Mouse

~~NT$1,449~~ NT$100

Add to cart

Razer Mamba Wireless Mouse

NT$1,990

Add to cart

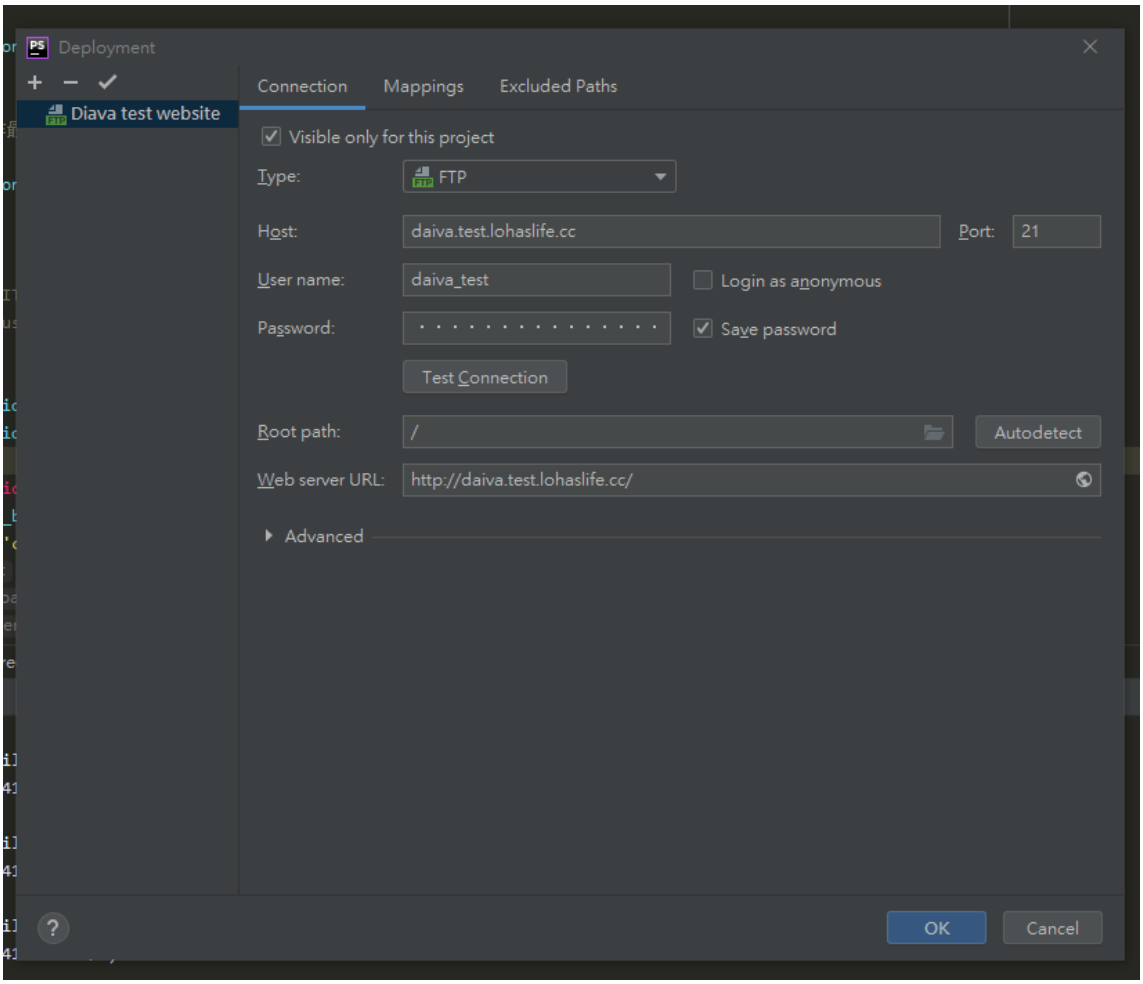
Logitech M175 Wireless Mouse

NT$435

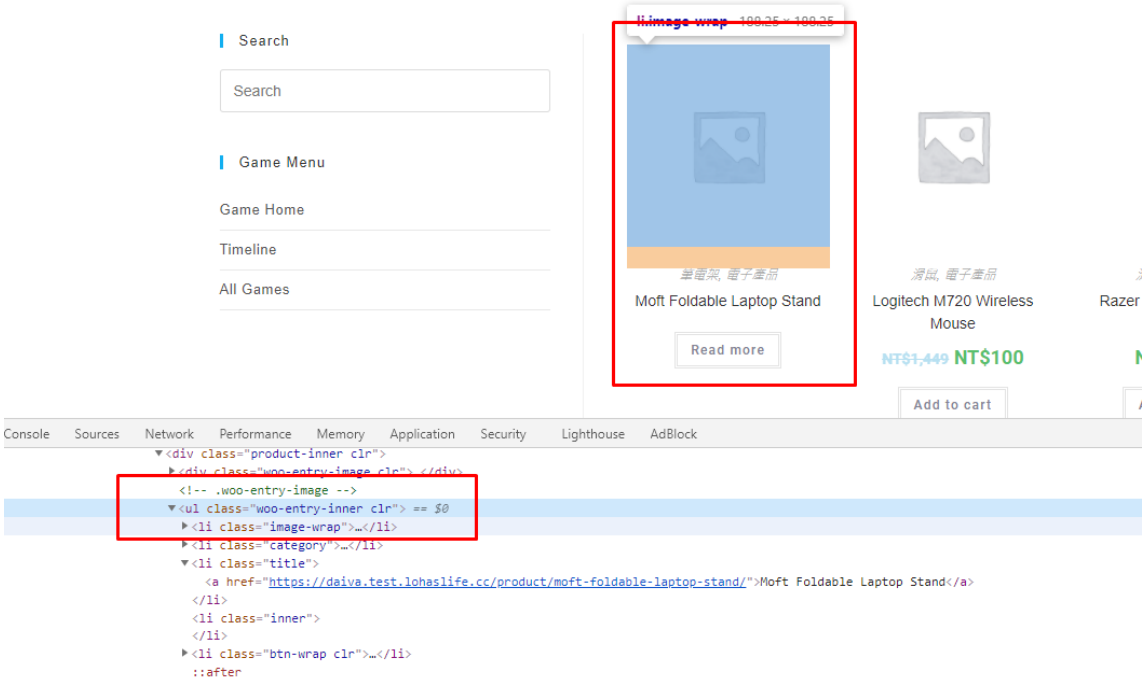
Add to cart

Backend Report

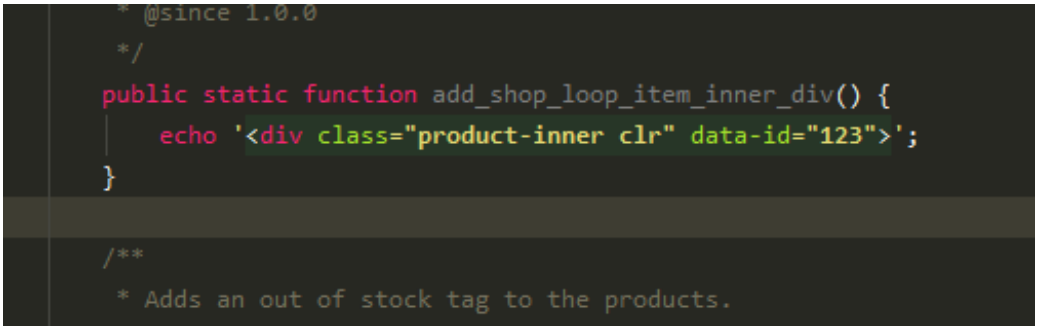
1. Connect your website with PS Code, by clicking Tools, Deployment, Configuration, Ok



2. Inspect element which you want to edit like below. (Find the class)

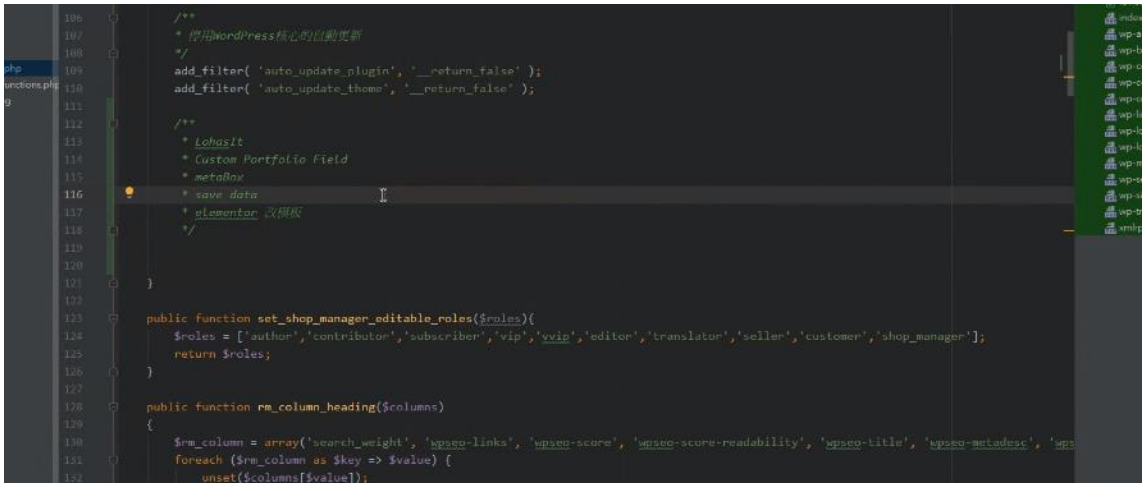


3. Try to add data-id="" to make sure the element.

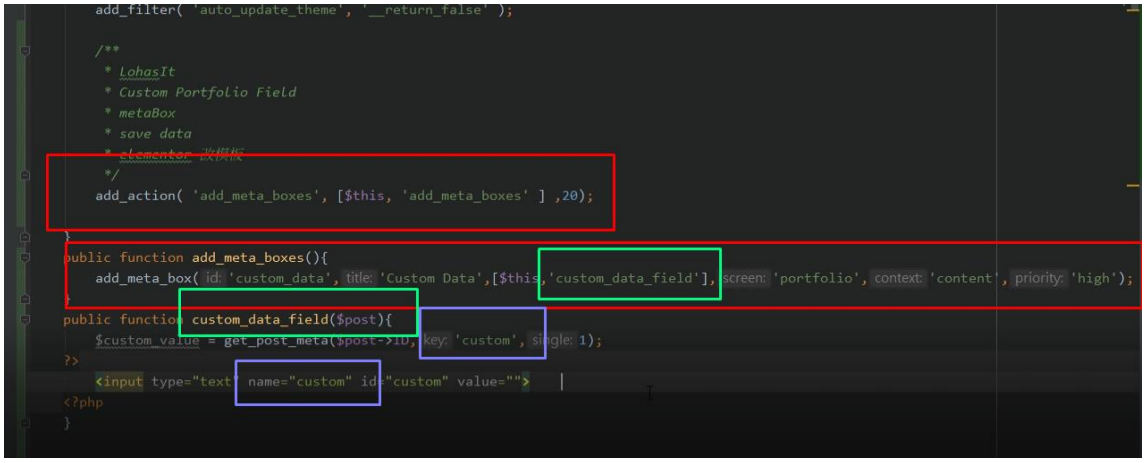


4. After you are sure, you may ctrl + click on the function in this example is

`add_shop_loop_item_inner_div()`



How to add metabox ?



5. In Han example, we want to modify the post that printed by plugins (sometimes can be printed by hooks). Try to modify metabox, metabox is a function that can help you to customize the pages, for example, title, function that we want to apply, category of the post, location, and priority.

```
if ( ! $query->found_posts ) {
    return;
}

$this->render_loop_header();

// It's the global `wp_query` it self. and the loop was started from the theme.
if ( $query->in_the_loop ) {
    $this->current_permalink = get_permalink();
    $this->render_post();
} else {
    while ( $query->have_posts() ) {
        $query->the_post();

        $this->current_permalink = get_permalink();
        $this->render_post();
    }
}
```

You may add $this->custom_portfolio().

```
        // It's the global `wp_query` it self. and the loop was started from the theme.
        if ( $query->in_the_loop ) {
            $this->current_permalink = get_permalink();
            $this->render_post();
        } else {
            while ( $query->have_posts() ) {
                $query->the_post();

                $this->current_permalink = get_permalink();
//              $this->render_post();
                $this->custom_portfolio();
            }
        }
```

Add the public function custom_portfolio() in the below of all function. Do not forget to add some title LohasIT, name of function, purpose. Copy all the function from protected function render_post()

```
    protected function render_post() {
        $this->render_post_header();
        $this->render_thumbnail();
        $this->render_text_header();
        $this->render_title();
        $this->render_meta_data();
        $this->render_excerpt();
        $this->render_read_more();
        $this->render_text_footer();
        $this->render_post_footer();
    }

    public function render_amp() {

    }

    /**
     * LohasIt
     * 修改模組 custom_portfolio 遊戲列表
     */
    public function custom_portfolio(){
        $this->render_post_header();
        $this->render_thumbnail();
        $this->render_text_header();
        $this->render_title();
        $this->render_meta_data();
        $this->render_excerpt();
        $this->render_read_more();
        $this->render_text_footer();
        $this->render_post_footer();
    }
}
```

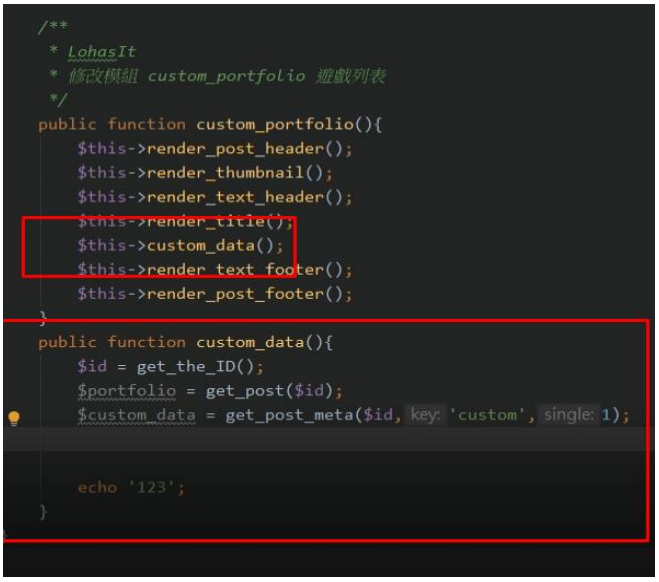
Delete and try to remake these functions:

`$this->render_meta_data().`

`$this->render_excerpt().`

`$this->render_read_more().`

In this example we want to get the ID of the products, get the post and some data from the post (ID selected).



```
/**
 * LohasIt
 * 修改模組 custom_portfolio 遊戲列表
 */
public function custom_portfolio(){
    $this->render_post_header();
    $this->render_thumbnail();
    $this->render_text_header();
    $this->render_title();
    $this->custom_data();
    $this->render_text_footer();
    $this->render_post_footer();
}
public function custom_data(){
    $id = get_the_ID();
    $portfolio = get_post($id);
    $custom_data = get_post_meta($id, key: 'custom', single: 1);


    echo '123';
}
```

Go to lohasit-custom.php (?) Not sure about this part, I think this file has been made before since the file name is lohasit-custom.php (?)

```
?>
    <input type="text" name="custom" id="custom" value="<?=$custom_value?>">
<?php
}
public function save_data_for_custom_data($post_id){
    if(isset($_POST['custom'])){
        update_post_meta($post_id, meta_key: 'custom', $_POST['custom']);
    }
}
```

Now, we want to print the html code like below.

```
}
public function custom_data(){
    $id = get_the_ID();
    $portfolio = get_post($id);
    $custom_data = get_post_meta($id, key: 'custom', single: 1);
    ?>
    <div style="background-color: red">
    <?=$custom_data?>
    </div>
    <?php

}
```

Now, in lohasit-custom.php, we want to print the price.

```
}
public function add_meta_boxes(){
    add_meta_box( id: 'custom_data', title: 'Custom Data',[$this,'custom_data_field'], screen: 'portfolio');
}
public function custom_data_field($post){
    $custom_value = get_post_meta($post->ID, key: 'custom', single: 1);
    $price = get_post_meta($post->ID, key: 'price', single: 1);
?>
    <input type="text" name="custom" id="custom" value="<?=$custom_value?>">
    <label for="price">Price</label>
    <input type="text" name="price" id="price" value="<?=$price?>">

<?php
}
public function save_data_for_custom_data($post_id){
    if(isset($_POST['custom'])){
        update_post_meta($post_id, meta_key: 'custom', $_POST['custom']);
    }
}
```

```
public function save_data_for_custom_data($post_id){
    if(isset($_POST['custom'])){
        update_post_meta($post_id, meta_key: 'custom', $_POST['custom']);
    }
    if(isset($_POST['price'])){
        update_post_meta($post_id, meta_key: 'price', $_POST['price']);
    }
}
```

Now, we want to add save_post in our file.

```php
    /**
     * LohasIt
     * Custom Portfolio Field
     * metaBox
     * save data
     * elementor 改模板
     */
    //metaBox
    add_action( 'add_meta_boxes', [$this, 'add_meta_boxes' ] ,20);
    //save_post
    add_action( 'save_post', [$this, 'save_data_for_custom_data'],1,2);


}
public function add_meta_boxes(){
    add_meta_box( id: 'custom_data', title: 'Custom Data',[$this,'custom_data_field'], screen: 'portfolio');
}
public function custom_data_field($post){
    $custom_value = get_post_meta($post->ID, key: 'custom', single: 1);
?>
    <input type="text" name="custom" id="custom" value="<?=$custom_value?>">
<?php
}
public function save_data_for_custom_data($post_id){
    if(isset($_POST['custom'])){
        update_post_meta($post_id,  meta_key: 'custom', $_POST['custom']);
    }
}
```

*custome data field is used to insert the html file, customize the layout.

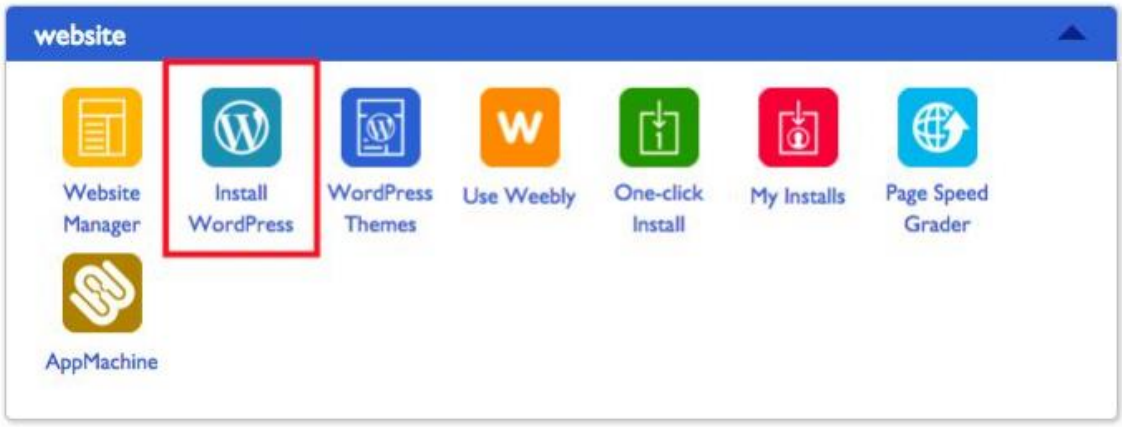
* in this part, add_meta_boxes and custom_data_field function has been made before in add meta box part (Part 1)

Day 3 - Thursday, 3rd December 2020
The Coding

Section 1 - File Structure

https://www.malcare.com/blog/beginners-guide-to-understanding-the-structure-of-a-wordpress-site/

After we install the wordpress, it will have two things happens, 1. Files 2. Database



Wordpress can be divided into 3 parts, such as codes, uploads, and configurations.

Part 1 - Codes

Codes can be divided into 3 parts, such as Core, Plugins, and Themes to create our site.

1. Core (The heart of WordPress)

Divided into 3 core files, such as wp-admin, wp-content, and wp-includes. (Be careful to modify this core, it can break our site)

2. Plugins and Themes help to design our site. A file named plugin will store all of our installed plugin. Themes are stored in a file called Themes.

Part 2 - Uploads

If we write a post with images, the imaes are uploaded and stored as a file, and the file called Uploads.

Part 3 - Configurations

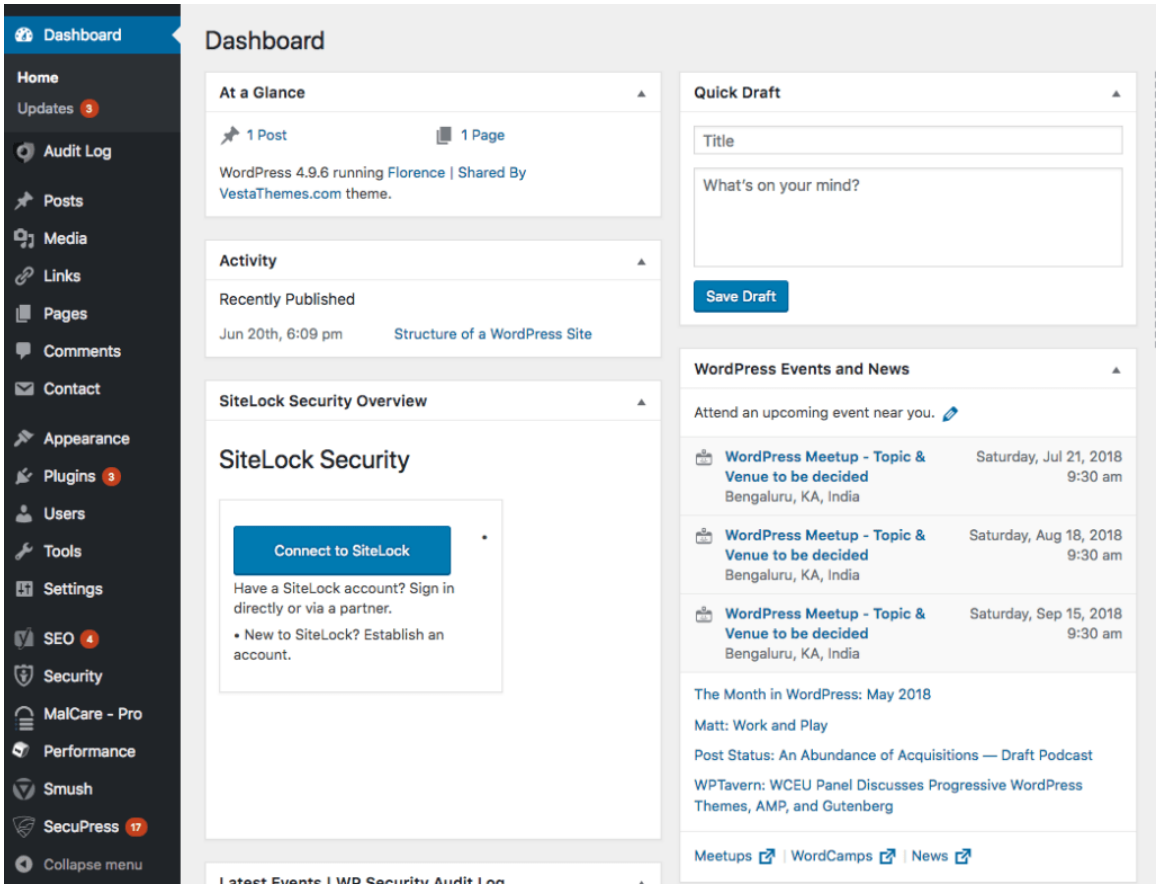
Includes configuration files like wp-config, it helps to connect the files to the database.

*To access our site files, we may visit our web host account, log in to our web host and go to a page called cPanel. We may select FileManager and see the structure will be like below.

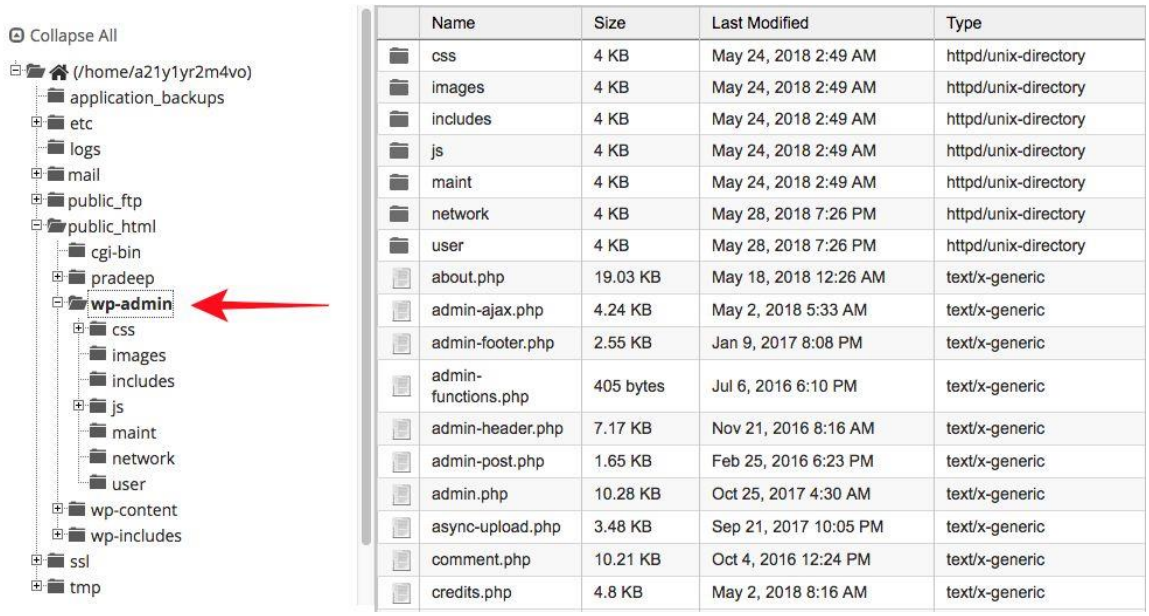
Core WordPress Files

1. wp-admin

Contains administrative files that power the WordPress dashboard. If we log in to our WordPress site, the first thing that we see is WordPress dashboard.

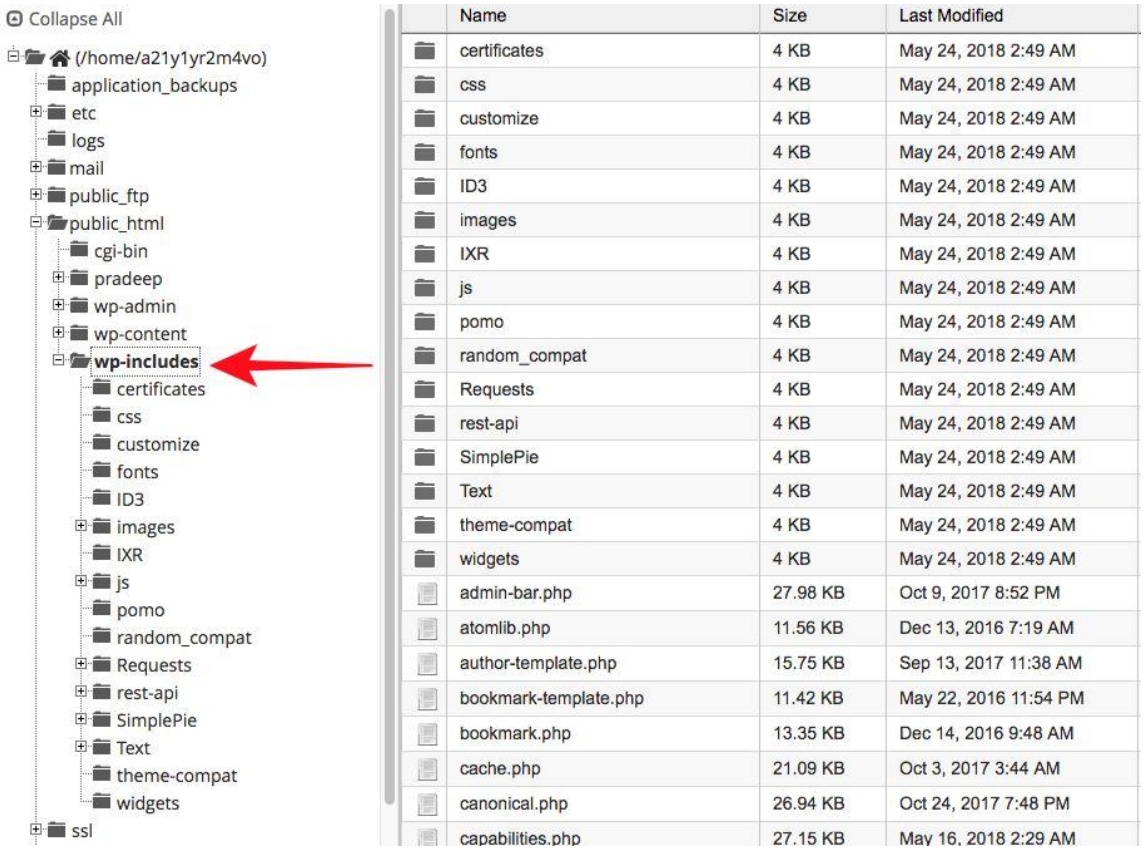


WordPress will check whether the credentials that you have provided are correct, or whether you are an admin or a simple contributor with limited access to the site.



2. wp-includes

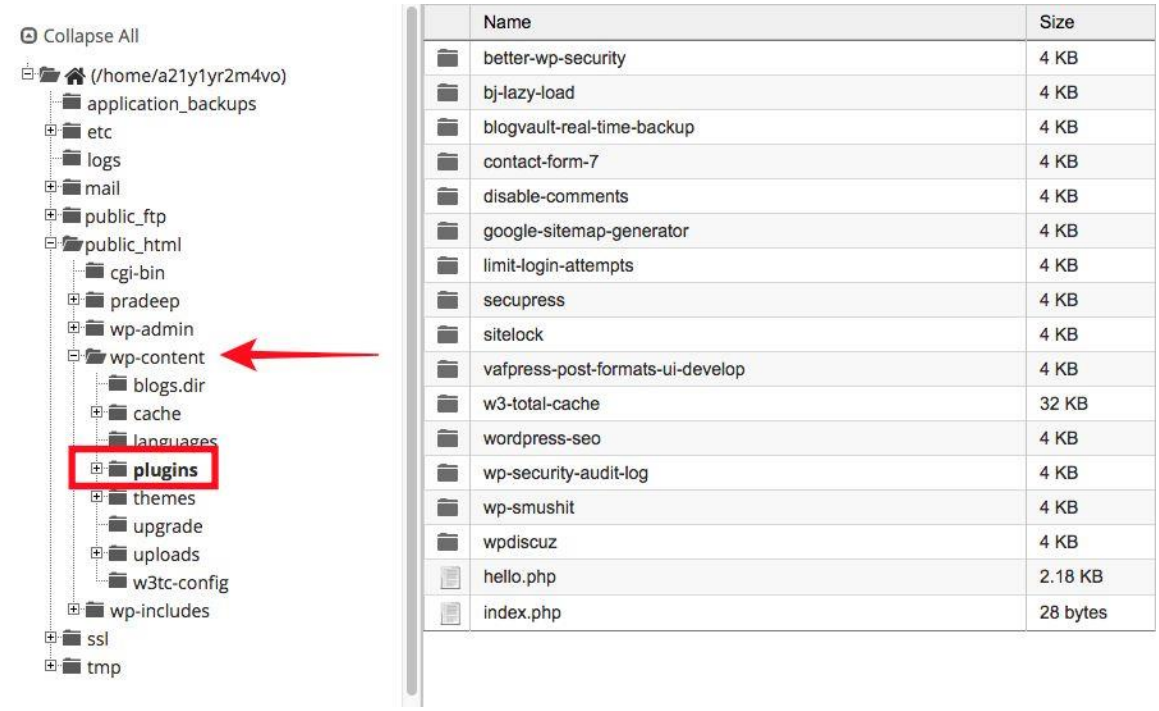
File in wp-includes will be responsible to determine how WordPress looks. This folder is large in size, and most of the WordPress core files are stored here. he texts that we see on WordPress, the font of the text, contain rules, hierarchies, and action command for some of the WordPress features.

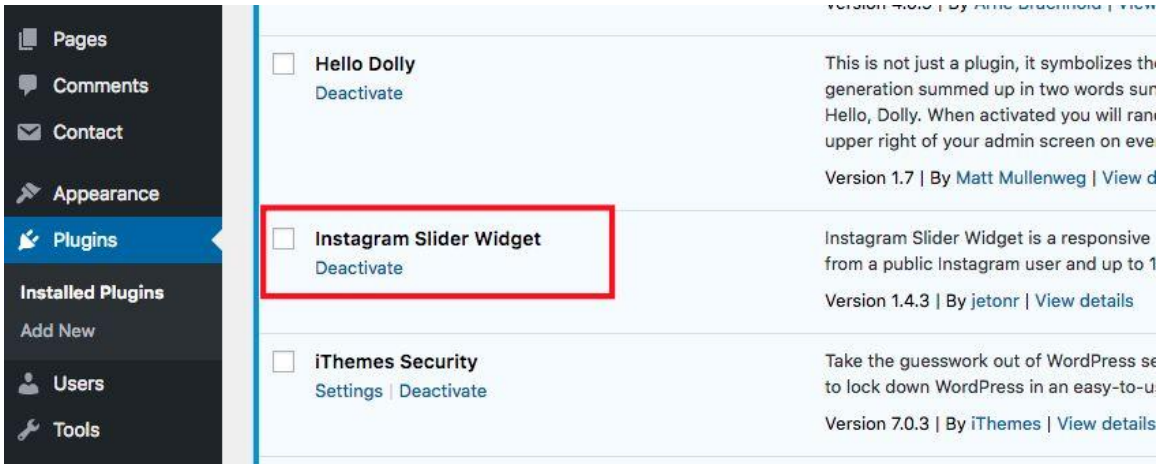


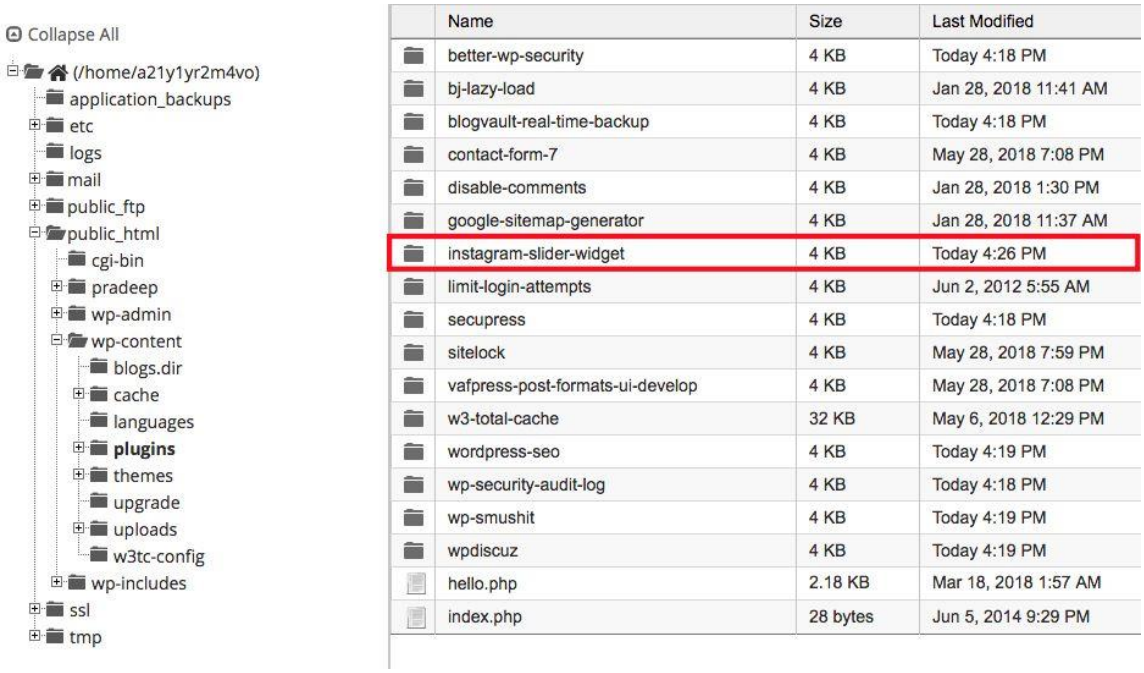
3. wp-content

Contains themes, plugins, and other uploads are stored.

Folder Plugin

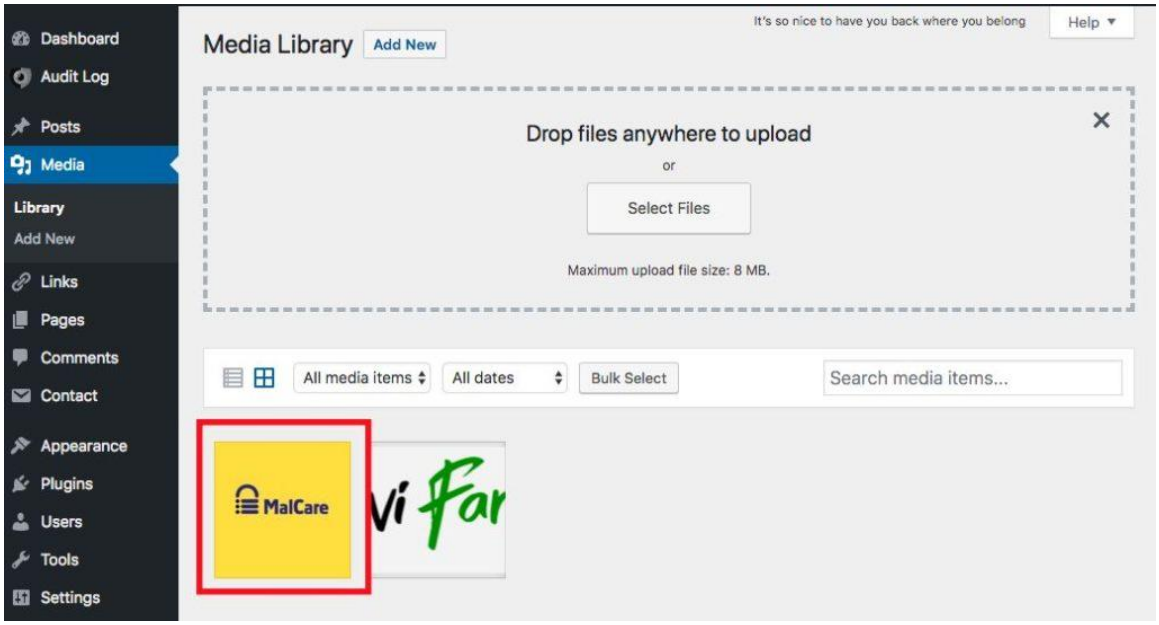


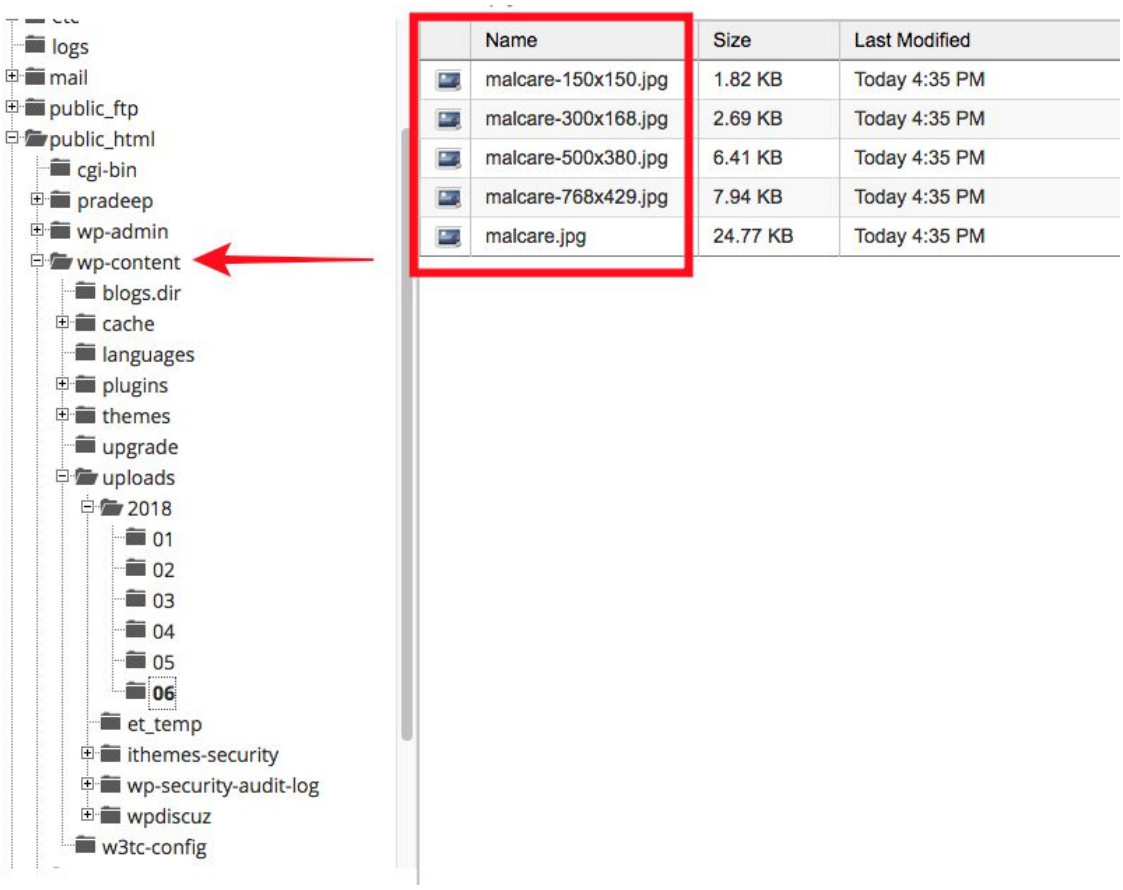




Folder Upload

Everytime you upload a picture on your WordPress site, it goes into the Upload folder.

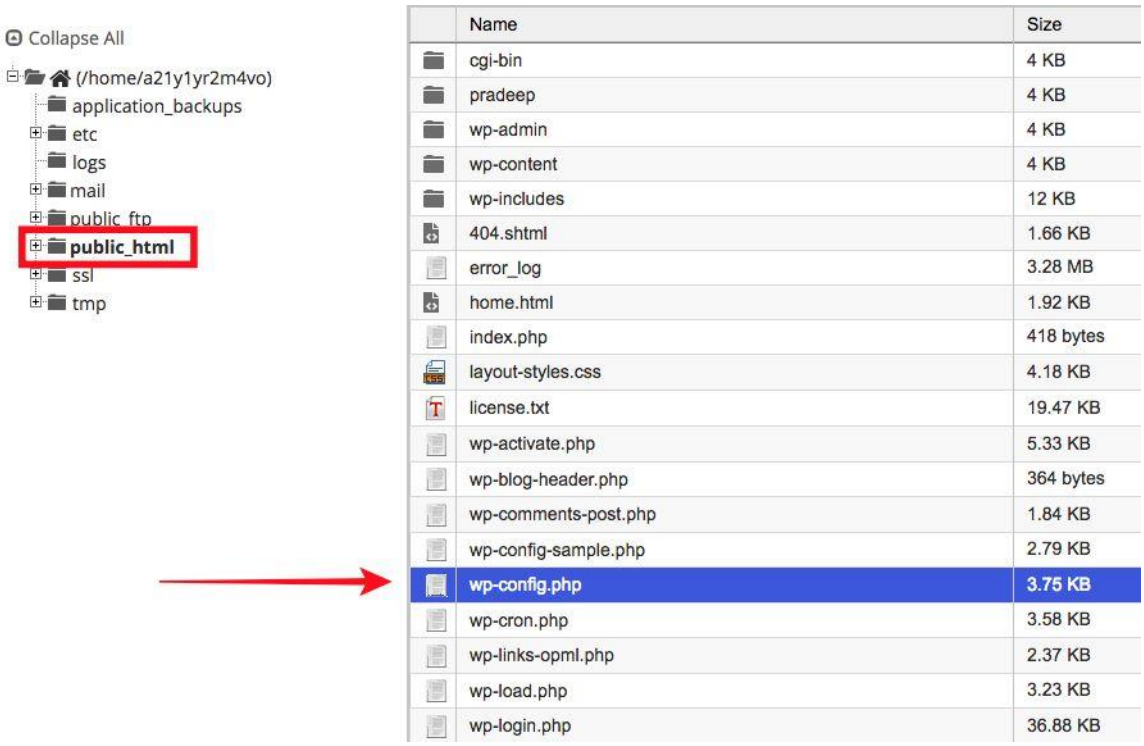




In **wp-admin, wp-content, and wp-includes** folders are often targeted by hackers. They hide malware in these folders mainly because most website owners do not often visit the File Manager or look into these folders.

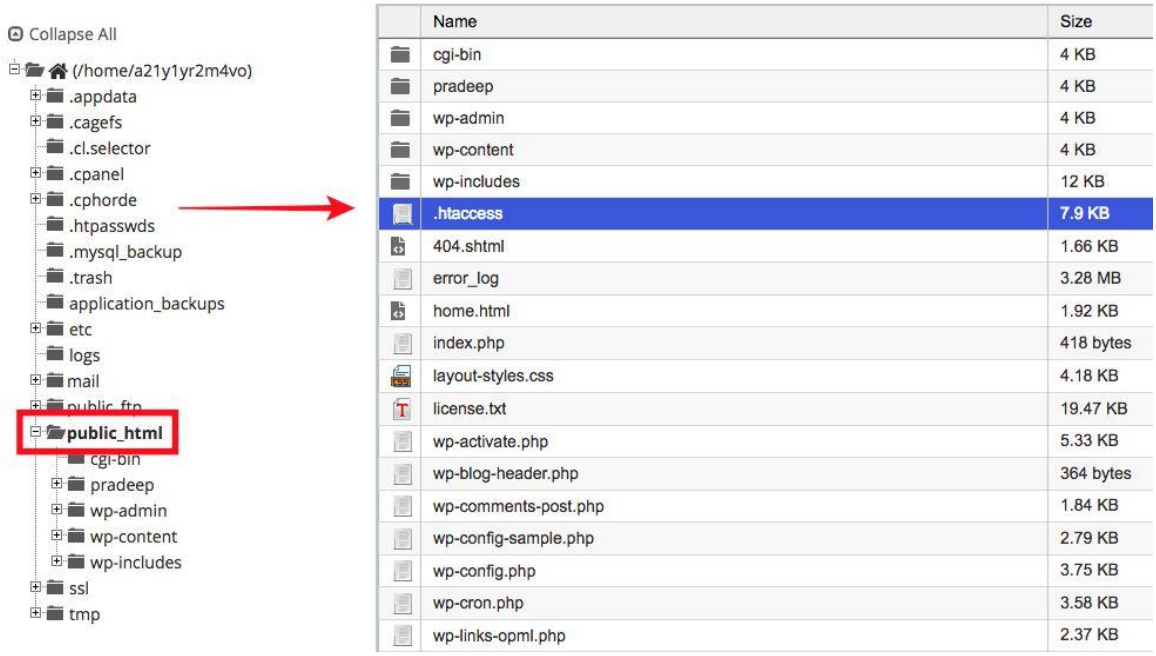
Hackers can keep utilizing a website as long as the website owner is unaware of the hack. Once a site owner finds out that their site has been hacked, they will clean it which will block the hacker's access to the site. Fear of being found out too soon prompts hackers to hide their bad codes in places where people are unlikely to look into.

wp-config



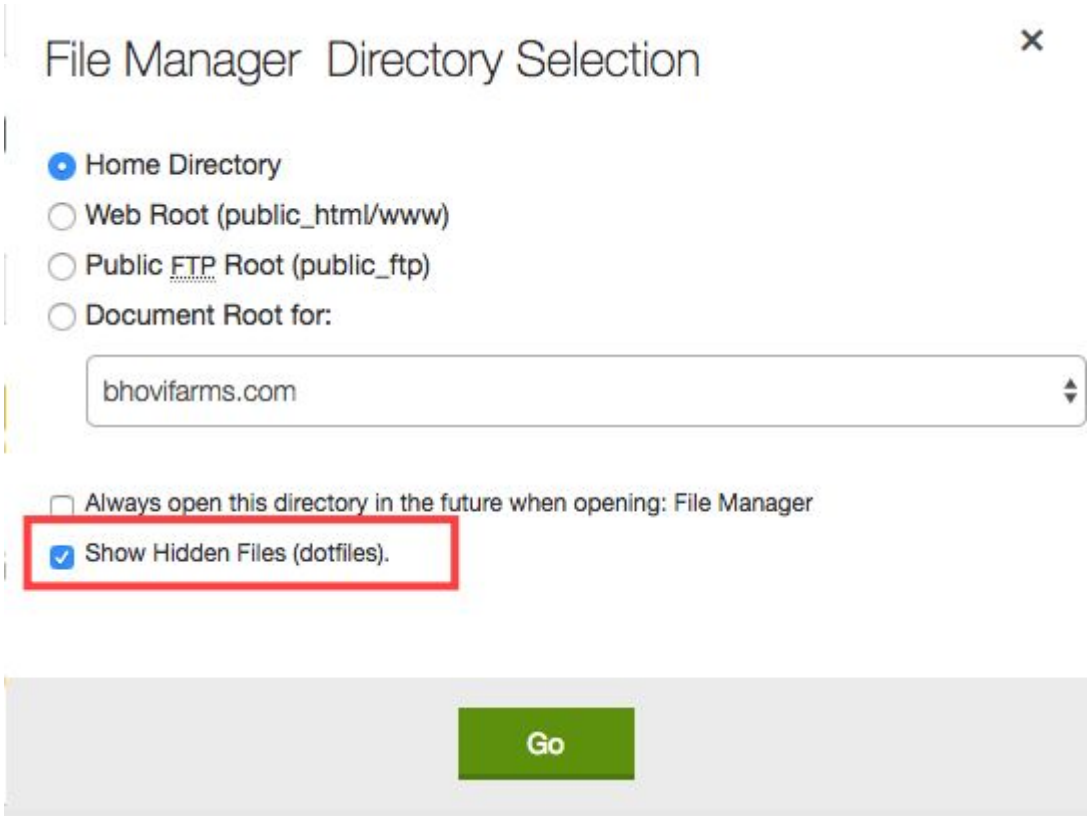
The wp-config file acts as a mediator between WordPress and the database. (Do not modify this file). **If wp-config is unable to connect your WordPress site with the database, your site will appear blank.**

.htaccess File



Using .htaccess you can make several changes to your WordPress site. It can help your password protect files and directory; it can prevent certain IP addresses from accessing your site, it can even help redirect visitors from one page to another, among other things.

Sometimes .htaccess is hidden and may not appear in the public_html folder. You need to go back to the cPanel and click on File Manager. A popup will appear where you will have to select 'Show Hidden Files.'



Database

Database in wordpress usually have 11 tables. Database tables have a default prefix wp_ which can be changed at will. These tables can be optimized and repaired. To access your database, you will have to visit your web host account. Log in to your web host and go to a page called cPanel. There, you should be able to find an option for phpMyAdmin. Select that, and a page will open that would look something like this:

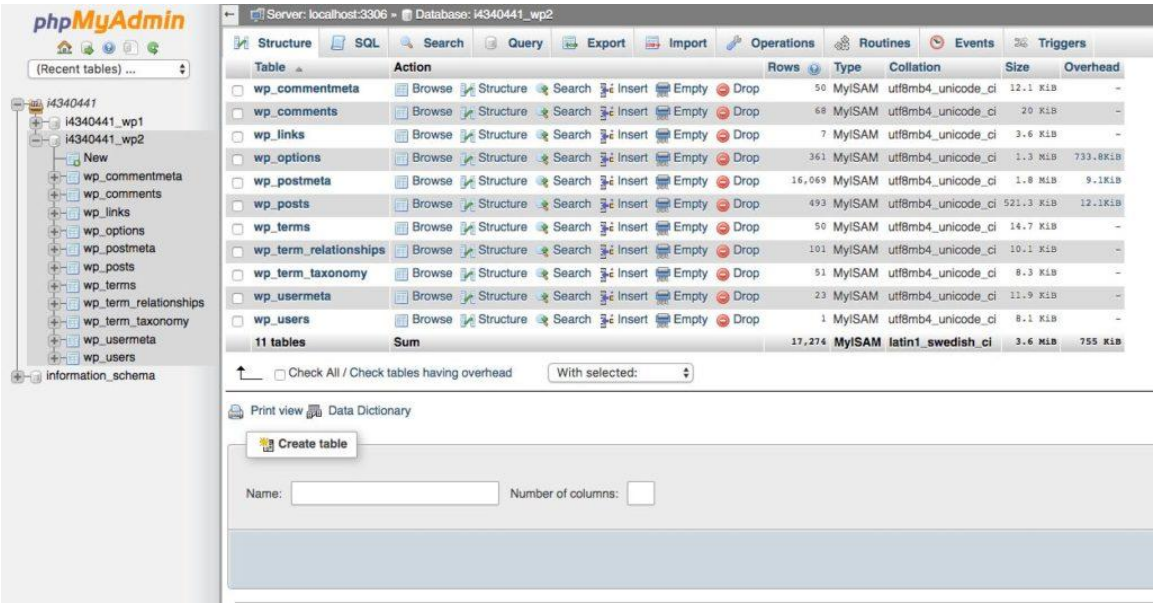


Table 1: wp_commentmeta

To store metadata for comments left on our Wordpress site. Information on things like whether a comment is approved or pending or trashed are stored in this table.

Table 2: wp_comments

To store information surrounding comments left on your account. It includes unique number assigned to each comment, email addresses, IP Addresses and URL for the comment author, pingback or trackback, replies to comments, time and date of the comment, among other things.

Table 3: wp_links

To take care of blogrolls (a list of hyperlinks to other blogs or websites) that was fashionable in the past. If someone is still using older versions of WordPress that has blogrolls, this particular table will help taking care of the blogroll feature.

Table 4: wp_options

To store data related to that Settings feature in our WordPress dashboard. If you select Setting from your WordPress dashboard, you will see options that enable you to change site title, tagline, site address (URL), etc.

Table 5: wp_postmeta

To store metadata (like Post ID, meta-ID, among other things) for posts and pages. Identification numbers allotted to each post helps to store them in a structured manner. The IDs also makes it easier to find posts and web pages when they are needed.

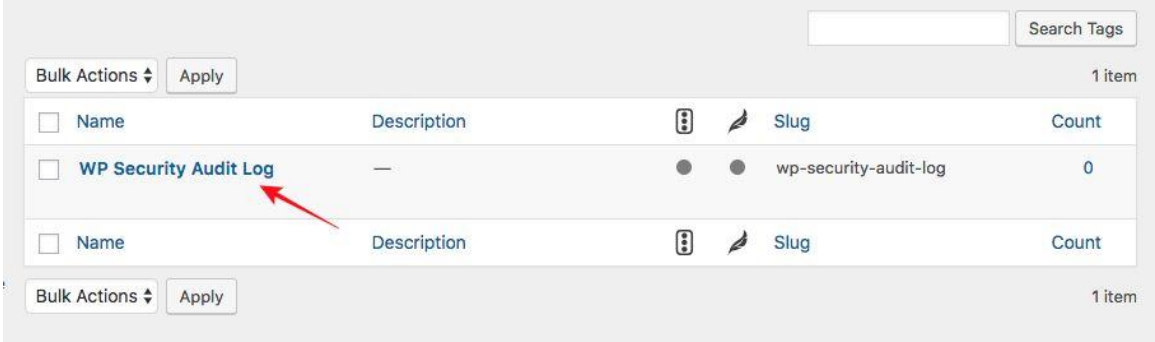
Table 6: wp_posts

To store information from posts, pages, and the navigation menu. In wp_postmeta, you will find unique identification number allotted to posts and pages whereas in wp_posts consist of information like post name, author name, postdate among other things.

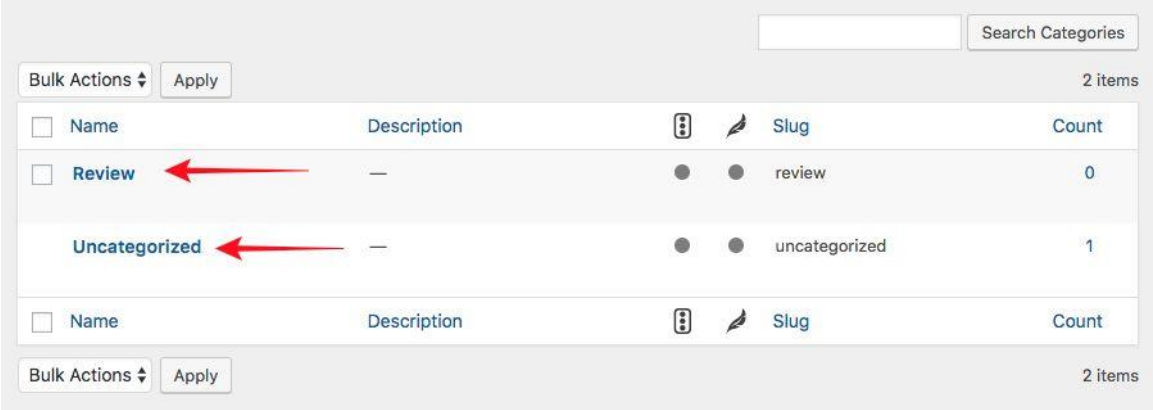
Table 7: wp_terms

To store three things: categories for tags of posts, categories of posts and link categories. To understand what these things are, let us take a look at the pictures below:

Tags of posts



Post Categories



Link Categories

In the three pictures above, we have:

- WP Security Audit Log (tag)
- Review (post categories)
- Uncategorized (post categories)
- Blogroll (link categories)

These tags, post categories, and link categories are present in the wp_terms table of my database.

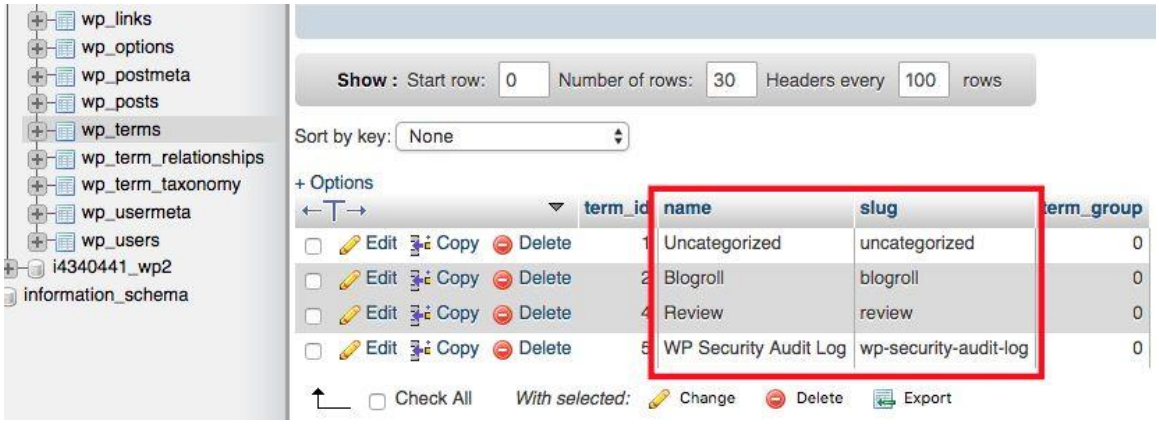


Table 8: wp_term_relationships

To store relationship data for categories and tags from the wp_terms table. If there is a post on your website named A. It belongs to the category 9. This table helps determine that post A belongs to 9 categories and not some other category.

Table 9: wp_term_taxonomy

To store descriptions of the taxonomy (tag, link, or category) for the entries in the wp_terms table. This table helps differentiate between tags, or links, or categories.

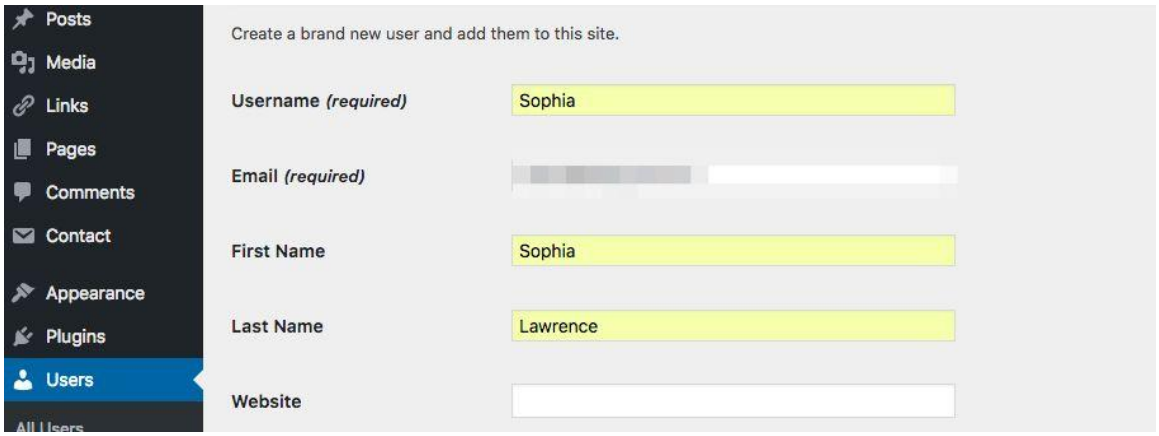
Table 10: wp_usermeta

To store metadata of a WordPress user. For instance, your user ID is 1, and your fellow admin's user ID is 2. This particular information is stored in the wp_usermeta table.

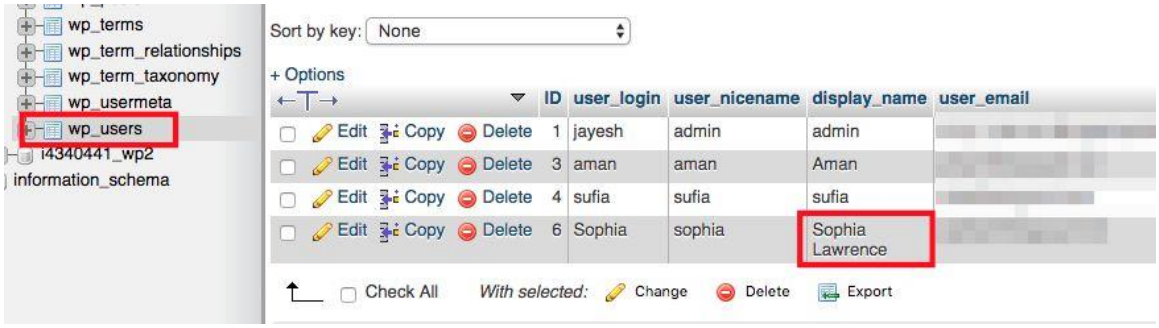
Table 11: wp_users

To store data for WordPress users. Well, the wp_usermeta table stored unique identification number of users but the wp_users table stores other user information like the Username, User Login Name, Email address, etc.

For example



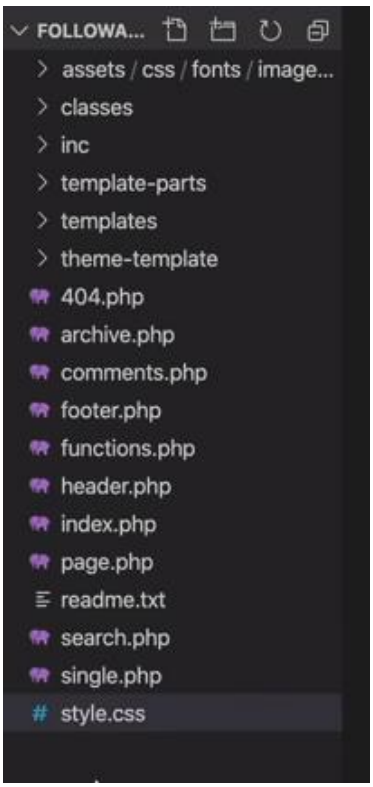
I looked up my wp_users table and saw information of the new user had been stored in there.



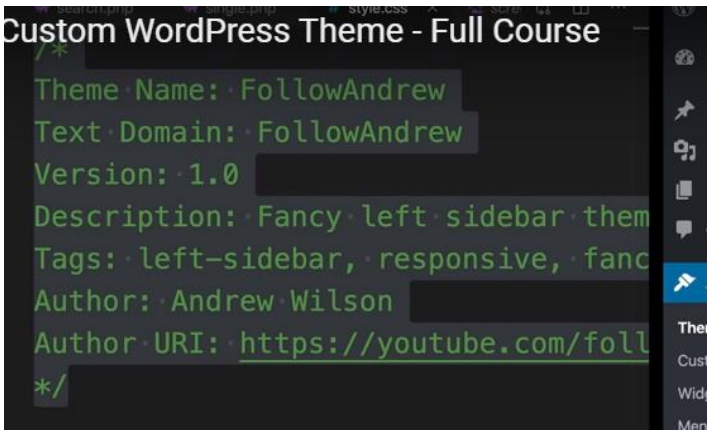
Section 2 - Theme Editing

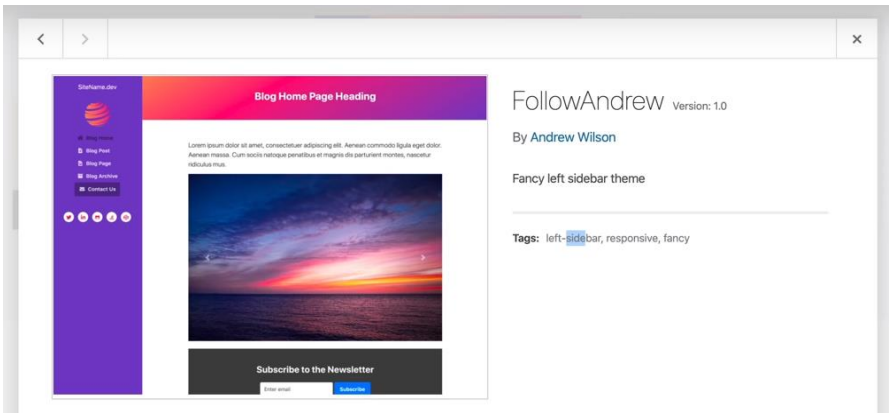
https://www.youtube.com/watch?v=-h7gOJbIpmo&ab_channel=freeCodeCamp.org

1. Prepare folder and file that you needed.



In style.css



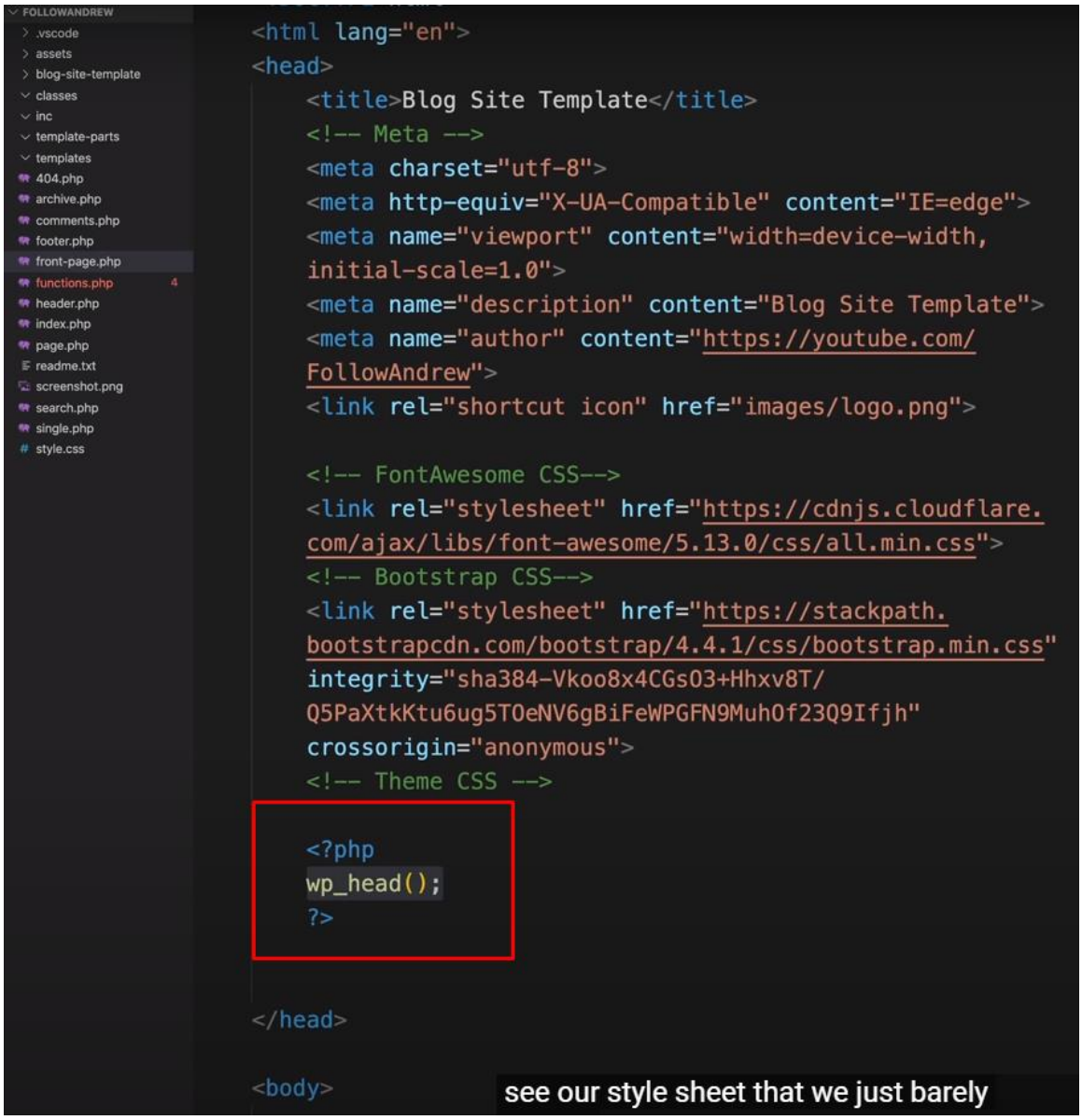


1. front-page.php

1. Copy index.html to front-page.php
2. Change the style css in header to our style.css like below

```
<!-- Theme CSS -->
<link rel="stylesheet" href="/wp-content/themes/followandrew/style.css">
```

2. front-page.php



```
            window._wpemojiSettings = {"baseUrl
            !function(e,a,t){var r,n,o,i,p=a.cr
        </script>
        <style type="text/css">
img.wp-smiley,
img.emoji {
    display: inline !important;
    border: none !important;
    box-shadow: none !important;
    height: 1em !important;
    width: 1em !important;
    margin: 0 .07em !important;
    vertical-align: -0.1em !important;
    background: none !important;
    padding: 0 !important;
}
</style>
    <link rel='stylesheet' id='wp-block-library
<link rel='stylesheet' id='followandrew-bootstr
<link rel='https://api.w.org/' href='http://wor
<link rel="EditURI" type="application/rsd+xml"
<link rel="wlwmanifest" type="application/wlwma
<meta name="generator" content="WordPress 5.3.2
```

3. functions.php (followandrew_register_styles will be called by wp_head() for style)

```php
function followandrew_register_styles(){

    wp_enqueue_style('followandrew-style', get_template_directory_uri() . "/style.css", array(), '1.0', 'all');
    wp_enqueue_style('followandrew-bootstrap', "https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/
    bootstrap.min.css", array(), '1.0', 'all');
    wp_enqueue_style('followandrew-fontawesome', "https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.13.0/
    css/all.min.css", array(), '1.0', 'all');

}

add_action( 'wp_enqueue_scripts', 'followandrew_register_styles');


?>
```

4. We want to make our theme is dynamic, so we must declare this in functions.php

```php
function followandrew_register_styles(){

    $version = wp_get_theme()->get( 'Version' );
    wp_enqueue_style('followandrew-style', get_template_directory_uri() . "/style.
    css", array('followandrew-bootstrap'), $version, 'all');
    wp_enqueue_style('followandrew-bootstrap', "https://stackpath.bootstrapcdn.com/
    bootstrap/4.4.1/css/bootstrap.min.css", array(), '4.4.1', 'all');
    wp_enqueue_style('followandrew-fontawesome', "https://cdnjs.cloudflare.com/ajax/
    libs/font-awesome/5.13.0/css/all.min.css", array(), '5.13.0', 'all');

}

add_action( 'wp_enqueue_scripts', 'followandrew_register_styles');


?>
```

5. Front-page.php (will be called by wp_footer() for javascript)

```php
<?php
    wp_footer();
?>
<!-- Bootstrap Javascript -->
<script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J6qa4849blE2
+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
crossorigin="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"
integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6"
crossorigin="anonymous"></script>
<script src="js/main.js"></script>
```

```
                </div>
            </div>
        </div>
        </article>
        <footer class="footer text-center py-2 theme-bg-dark">

            <p class="copyright"><a href="https://youtube.com/FollowAndrew">FollowAndrew</a></p

        </footer>

    </div>

    <script type='text/javascript' src='http://wordpress.mac/wp-i
    <!-- Bootstrap Javascript -->
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0
    <script src="https://stackpath.bootstrapcdn.com/bootstr
    <script src="js/main.js"></script>

</body>
</html>
```

```
function followandrew_register_scripts(){


    wp_enqueue_script('followandrew-jquery','https://code.jquery.com/jquery-3.4.1.slim.min.js', array(), '3.4.
    1',true);
    wp_enqueue_script('followandrew-popper','https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.
    js', array(), '1.16.0',true);

    wp_enqueue_script('followandrew-bootstrap','https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.
    min.js', array(), '3.4.1',true);

    wp_enqueue_script('followandrew-jquery',get_template_directory_uri()."/assets/js/main.js", array(), '1.0',
    true);


}

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

6. If we want to handle our title of pages automatifally, the way like below.

7. Ceate main.css in assets/js/main.css

8. followandrew_theme_support() is used to call support functions (not header not footer, for example title tag. The example is like below.

```
function followandrew_theme_support(){
    // Adds dynamic title tag support
    add_theme_support('title_tag');
}
```

functions.php | main.js | front-page.php | footer.php | header.php

```php
<?php

function followandrew_theme_support(){
    // Adds dynamic title tag support
    add_theme_support('title_tag');
}

add_action('after_setup_theme','followandrew_theme_support');
```

9. Make a dynamic content in front-page.php

Move header and footer to their corresponding files, then create a content with dynamic post handled by wordpress.

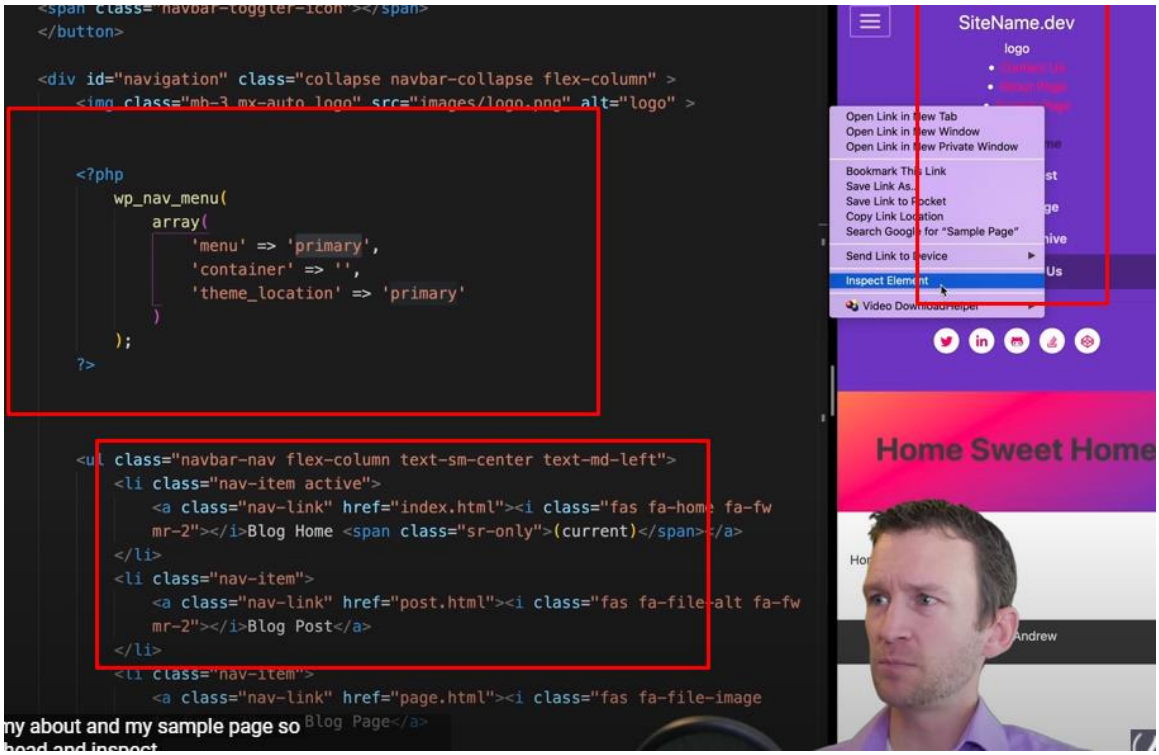
```php
<?php
 get_header();
?>

    <article class="content px-3 py-5 p-md-5">


    <?php
        if( have_posts() ){

            while( have_posts() ){

                the_post();
                the_content();
            }

        }
    ?>


    </article>


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

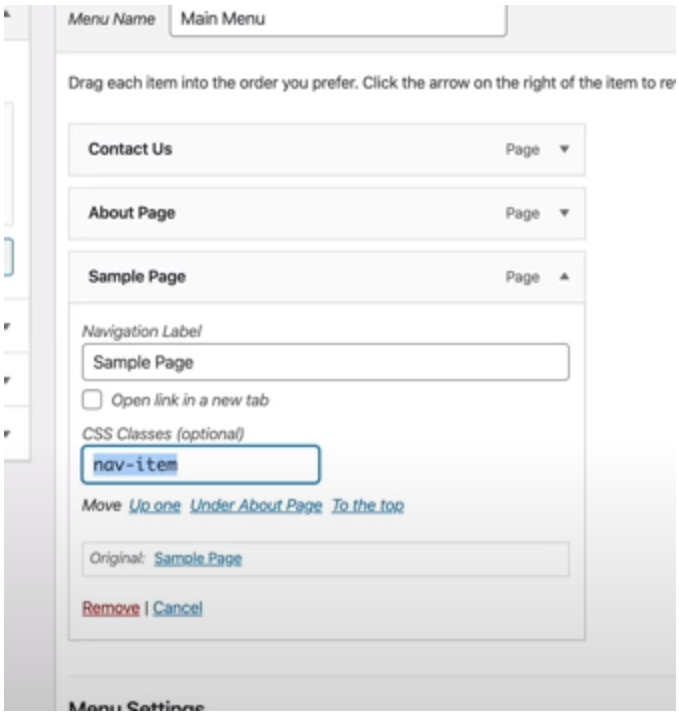
10. In functions.php we make our own menu, such as side bar menu. (or burger menu)

1. In functions.php we may prepare our customize menu like below.

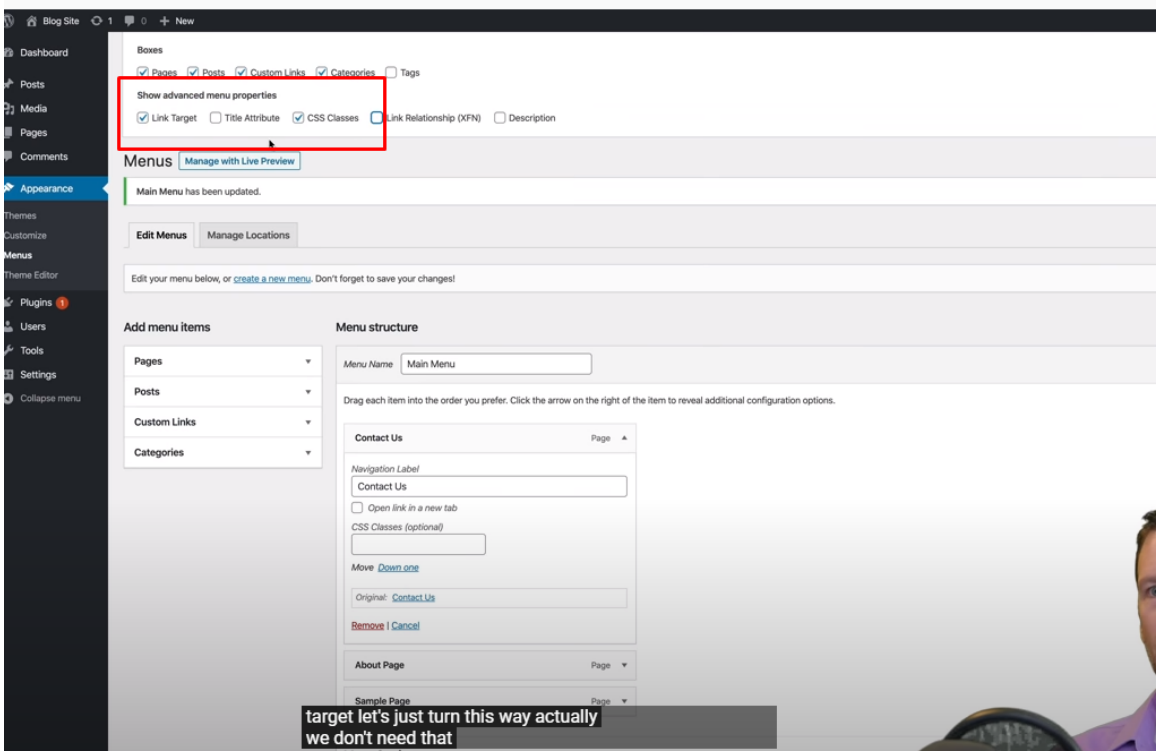
```php
function followandrew_menus(){

    $locations = array(
        'primary' => "Desktop Primary Left Sidebar",
        'footer' => "Footer Menu Items"
    );

    register_nav_menus($locations);
}

add_action('init','followandrew_menus');
```

2. In header.php we may remove all the menu and change it like the code below.





3. Turn of the CSS classes.



===> Need to learn more, until 1:16:33

Section 3 - Woocommerce Shortcodes

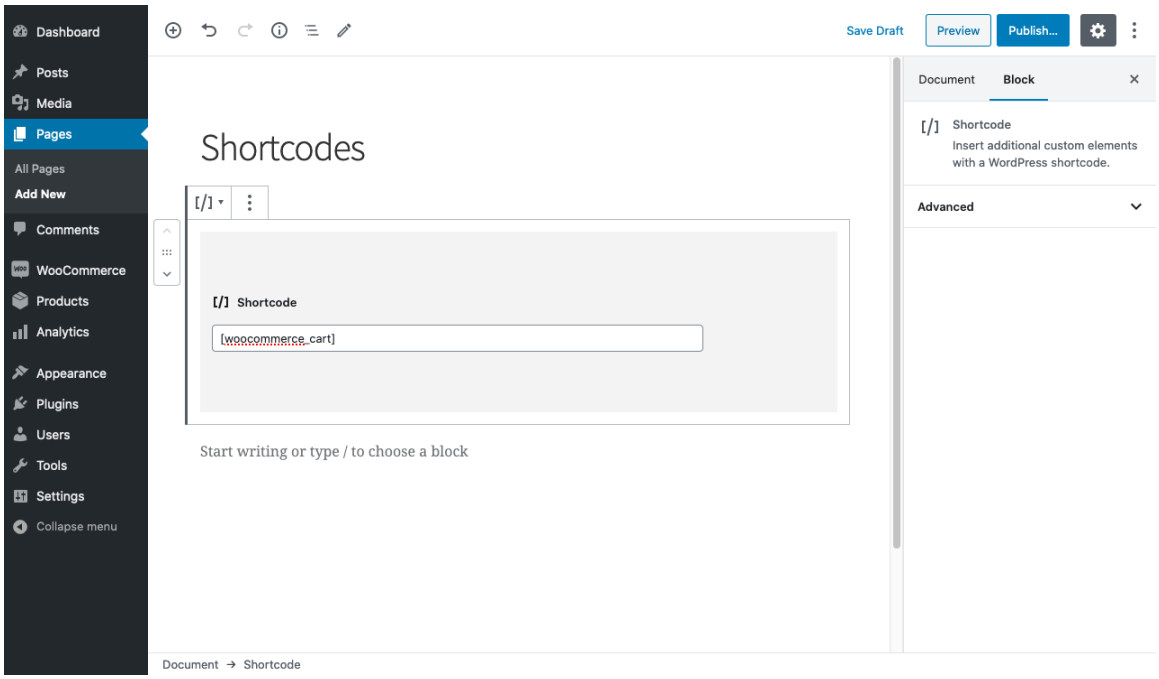
https://docs.woocommerce.com/document/woocommerce-shortcodes/

1. Introduction

Our WooCommerce blocks are now the easiest and most flexible way to display your products on posts and pages on your WooCommerce site.

2. Where to use

Shortcodes can be used on pages and posts in WordPress. If you are using the block editor, there is a shortcode block you can use to paste the shortcode in.



3. Page Shortcodes

WooCommerce cannot function properly without the first three shortcodes being somewhere on your site.

[woocommerce_cart] – shows the cart page

[woocommerce_checkout] – shows the checkout page

[woocommerce_my_account] – shows the user account page

[woocommerce_order_tracking] – shows the order tracking form

1. Cart

To make a cart page.

[woocommerce_cart]

2. Checkout

To make a checkout page.

[woocommerce_checkout]

3. My Account

To make customer accoutn page, which can view past orders, their information.

```
array(
    'current_user' => ''
 )
```

[woocommerce_my_account]

*Note:

Current user argument is automatically set using get_user_by('id', get_current_user_id()).

4. Order Tracking

To make a tracking form, let the user see the status of an order by entering their order detials.

[woocommerce_order_tracking]

5. Products

The [products] shortcode allows you to display products by post ID, SKU, categories, attributes, with support for pagination, random sorting, and product tags, replacing the need for multiples shortcodes such as [featured_products], [sale_products], [best_selling_products], [recent_products], [product_attribute], and [top_rated_products], which are needed in versions of WooCommerce below 3.2. Review the examples below.

1. Display Product Attributes

- limit - number of products we want to display (-1 means display all)
- columns - number of columns to display (default 4)
- paginate - toggles pagination on.
- orderby - sort the products displayed by the entered option. (option: date, id, menu_order, popularity, rand, rating, title)
- skus - comma-separated list of products SKUs (?)
- category - comma-separated list of category slugs (?)
- tag - comma separated list of tag slugs (?)
- order - states whether the product order is ASC or DESC, using the method set in orderby/
- class - adds an HTML wrapper class , we may modifty the specifi output with custom CSS
- on_sale - show the sale products
- best_selling - show the best sellign products (on_sale, top_rated)
- top_Rated - show top rated products. (on_sale or best_selling)

2. Content Product Attributes

- attribute - show products using specified atribute slug
- terms - comma separated list of attribvute terms to be used with attribute (?)
- terms_operator - to compare attrbiutes terms, AND, IN, NOT IN
- tag_operator - to compare tags, AND, IN, NOT IN
- visibility - dipslay products based on the selected visibility, consists of
  - visible:h product will be visible on shop and search results. (default)
  - catalog: products visible on the shop only, but not search results
  - search: products visible in search results only, but not on the shop
  - hidden products are hidden from both shop and search, accessed by URL only
  - featured: products are marked as Featured Products
- category - show products using the specified category slug. (?)
- tag - show products using the specified tag slug. (?)
- cat_operator - to compare category terms, AND, IN, NOT IN
- ids - display products based on comma-separated list of Post IDs. (?)
- skus - display products based on comma-spearated list of SKUs. (?)

3. Special Product Attributes

These attributes cannot be used with the “Content Attributes” listed above, as they will likely cause a conflict and not display. You should only use one of the following special attributes.

- best_selling – Will display your best-selling products. Must be set to true.
- on_sale – Will display your on-sale products. Must be set to true.

4. Example Scenario

1. Scenario 1 - Random Sale Items

To display four random on sale products.

[products limit="4" columns="4" orderby="popularity" class="quick-sale" on_sale="true"]

2. Scenario 2 - Featured Products

To display my featured products, two per row, with a maximum of four items.

[products limit="4" columns="2" visibility="featured"]

3. Scenario 3 - Best Selling Products

To display my three top bestselling products in one row.

[products limit="3" columns="3" best_selling="true"]

4. Scenario 4 - Newest Products

To display the newest products first – four products across one row. To accomplish this, we will use the Post ID (which is generated when the product page is created), along with the order and orderby command. Since you cannot see the Post ID from the frontend, the ID#s have been superimposed over the images.

[products limit="4" columns="4" orderby="id" order="DESC" visibility="visible"]

5. Scenario 5 - Specific Categories

To display hoodies and shirts, but not accessories. I will use two rows of four.

[products limit="8" columns="4" category="hoodies, tshirts" cat_operator="AND"]

Alternatively, I only want to display products not in those categories. All I need to change is the cat_operator to NOT IN.
[products limit="8" columns="4" category="hoodies, tshirts" cat_operator="NOT IN"]

6. Scenario 6 – Attribute Display
Each of the clothing items has an attribute, either “Spring/Summer” or “Fall/Winter” depending on the appropriate season, with some accessories having both since they can be worn all year. In this example, I want three products per row, displaying all of the “Spring/Summer” items. That attribute slug is season, and the attributes are warm and cold. I also want them sorted from the newest products to the oldest.
[products columns="3" attribute="season" terms="warm" orderby="date"]

Alternatively, if I wanted to display exclusively cold weather products, I could add NOT IN as my terms_operator:
[products columns="3" attribute="season" terms="warm" terms_operator="NOT IN"]

7. Scenario 7 – Show Only Products With tag “hoodie”
[products tag="hoodie"]

5. Sorting Products by Custom Meta Fields

When using the Products shortcode, you can choose to order products by the pre-defined values above. You can also sort products by custom meta fields using the code below (in this example we order products by price):

```
add_filter( 'woocommerce_shortcode_products_query',
'woocommerce_shortcode_products_orderby' );

function woocommerce_shortcode_products_orderby( $args ) {

$standard_array = array('menu_order','title','date','rand','id');

    if( isset( $args['orderby'] ) && !in_array( $args['orderby'], $standard_array )){
        $args['meta_key'] = $args['orderby'];
        $args['orderby']  = 'meta_value_num';
    }
```

```
    return $args;
}
```

6. Product Category

1. Shortcode

[product_category] – Will display products in a specified product category.

[product_categories] – Will display all your product categories.

2. Available Product Category Attributes

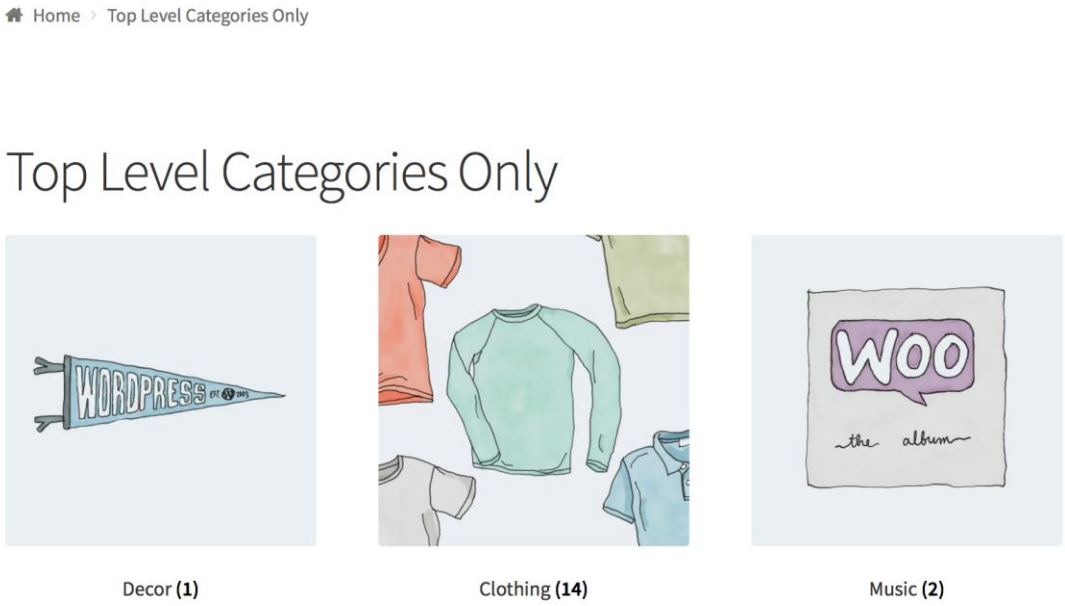
- ids – Specify specific category ids to be listed. To be used in [product_categories]
- category – category id, name or slug. To be used in [product_category]
- limit – show the number of categories to display
- columns – show the number of columns to display. Defaults to 4
- hide_empty – Will hide empty categories. The default is “1” which will hide empty categories. Set to “0” to show empty categories
- parent – Set to a specific category ID if you would like to display all the child categories. Alternatively, set to “0” (like in the example below) to show only the top-level categories. (?)
- orderby – The default is to order by “name”, can be set to “id”, “slug”, or “menu_order”. If you want to order by the ids, you specified then you can use orderby="include"
- order – States whether the category ordering is ascending (ASC) or descending (DESC), using the method set in orderby. Defaults to ASC.

3. Example of Product Category Scenarios

Scenario 8 – Show Top Level Categories Only

Imagine you only wanted to show top level categories on a page and exclude the sub categories, well it is possible with the following shortcode.

[product_categories number="0" parent="0"]



7. Product Page

Show a full single product page by ID or SKU. *SKU (?)

[product_page id="99"]

[product_page sku=”FOO”]

8. Related Products

List related products.

```
array(
    'limit' => '12',
    'columns' => '4',
    'orderby' => 'title'
 )
```

[related_products limit=”12″]

9. Add to Cart

Show the price and add to cart button of a single product by ID.

Args:

```
array(
    'id' => '99',
    'style' => 'border:4px solid #ccc; padding: 12px;',
    'sku' => 'FOO'
    'show_price' => 'TRUE'
    'class' => 'CSS-CLASS'
    'quantity' => '1';
)
[add_to_cart id="99"]
```

10. Add to Cart URL

Display the URL on the add to cart button of a single product by ID.

Args:

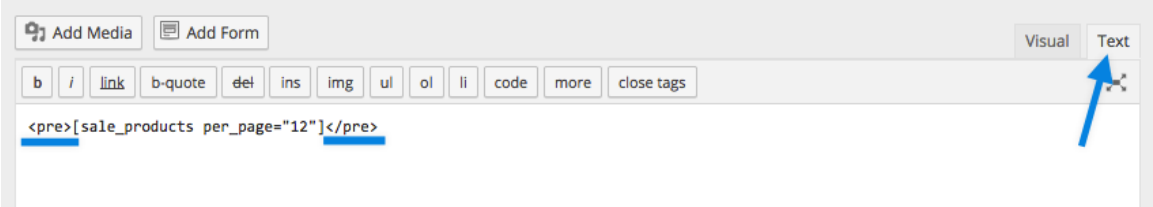
```
array(
    'id' => '99',
    'sku' => 'FOO'
)
[add_to_cart_url id="99"]
```

11. Display WooCommerce notifications on pages that are not WooCommerce

[shop_messages] allows you to show WooCommerce notifications (like, 'The product has been added to cart') on non-WooCommerce pages. Helpful when you use other shortcodes, like [add_to_cart], and would like the users to get some feedback on their actions.

12. Troubleshooting Shortcodes

If you correctly pasted your shortcodes and the display looks incorrect, make sure you did not embed the shortcode between <pre> tags. This is a common issue. To remove these tags, edit the page, and click the Text tab:



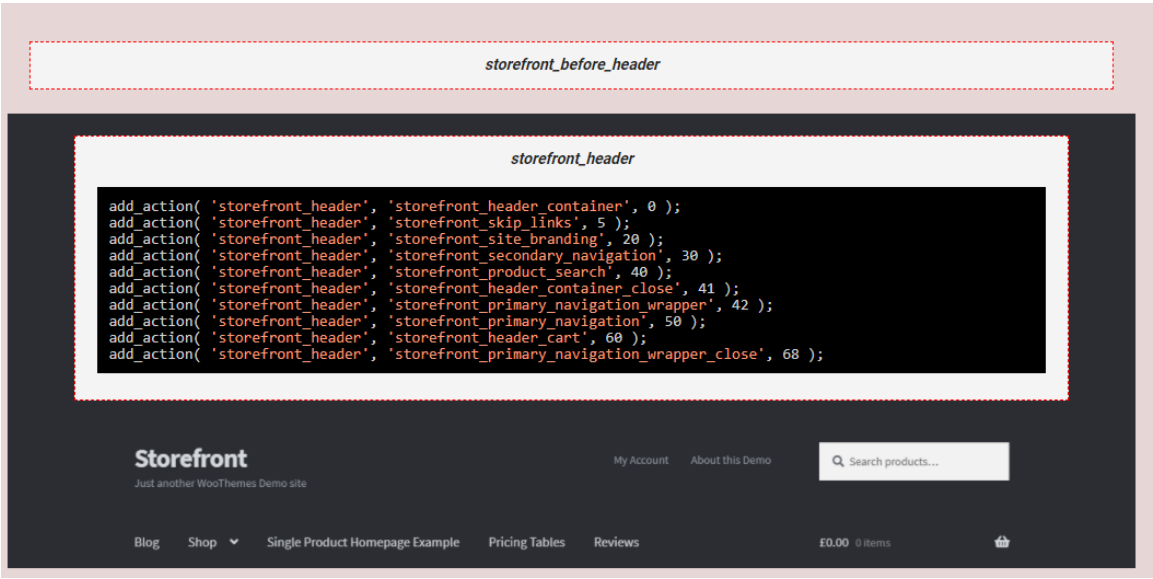
Another common problem is that straight quotation marks (") are displayed as curly quotation marks (“). For the shortcodes to work correctly, you need straight quotation marks.

Section 4 - Woocommerce Common Hooks - Visual Guide

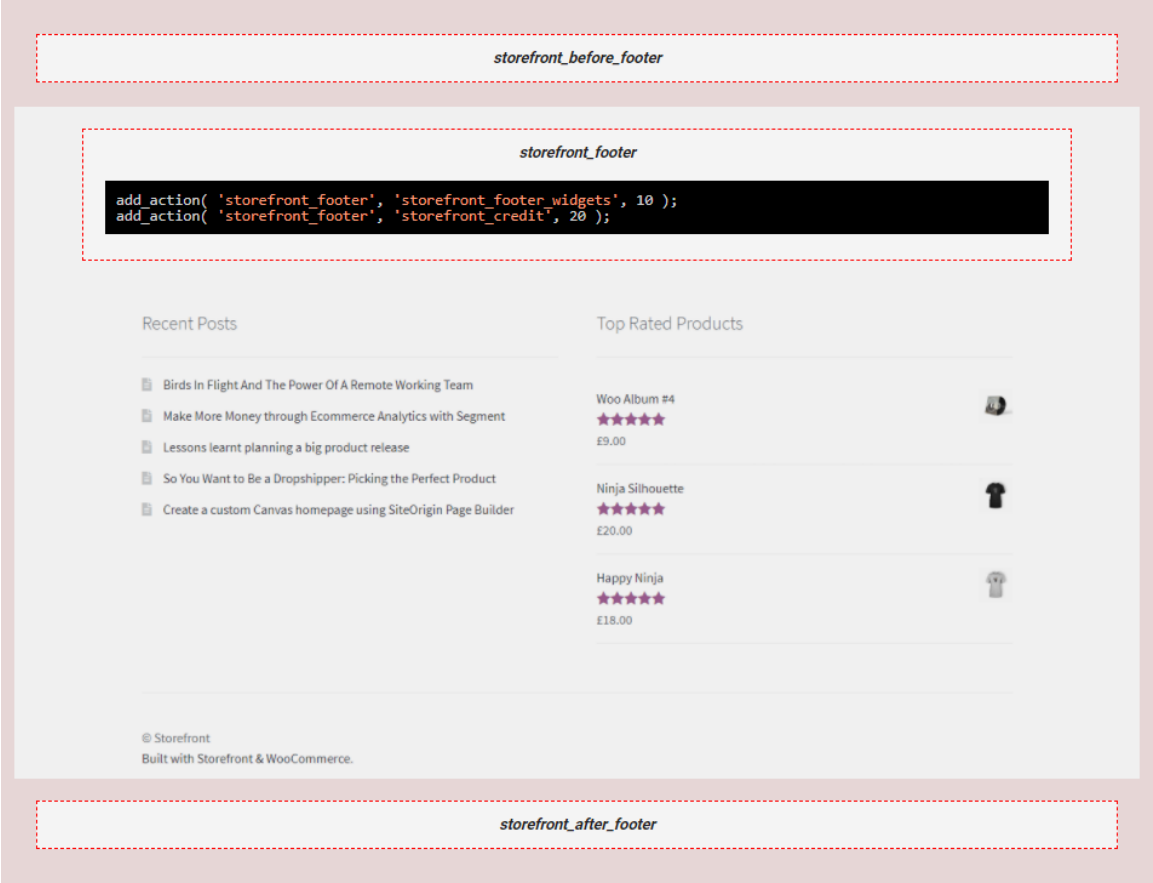
https://www.businessbloomer.com/category/woocommerce-tips/visual-hook-series/

https://www.businessbloomer.com/storefront-theme-visual-hook-guide/#more-20297

1. Storefront Theme > Header Hooks



2. Storefront Theme > Footer Hooks



3. Storefront Theme > Sidebar Hooks

```
add_action( 'storefront_sidebar', 'storefront_get_sidebar', 10 );
```

4. Storefront Theme > Standard Page Hooks

```
add_action( 'storefront_before_content', 'storefront_header_widget_region', 10 );
add_action( 'storefront_before_content', 'woocommerce_breadcrumb', 10 );
add_action( 'storefront_page', 'storefront_page_header', 10 );
add_action( 'storefront_page', 'storefront_page_content', 20 );
add_action( 'storefront_page_after', 'storefront_display_comments', 10 );
```

5. Storefront Theme > “Homepage Page Template” Hooks

storefront_before_content

```
add_action( 'storefront_before_content', 'storefront_header_widget_region', 10 );
add_action( 'storefront_before_content', 'woocommerce_breadcrumb', 10 );
```

homepage

```
add_action( 'homepage', 'storefront_homepage_content', 10 );
add_action( 'homepage', 'storefront_product_categories', 20 );
add_action( 'homepage', 'storefront_recent_products', 30 );
add_action( 'homepage', 'storefront_featured_products', 40 );
add_action( 'homepage', 'storefront_popular_products', 50 );
add_action( 'homepage', 'storefront_on_sale_products', 60 );
add_action( 'homepage', 'storefront_best_selling_products', 70 );
```

Home

Content of the WordPress page goes here.

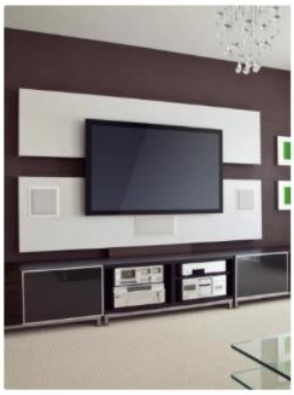
storefront_homepage_before_product_categories

Product Categories

storefront_homepage_after_product_categories_title

Clothing (46)

Kitchen (28)



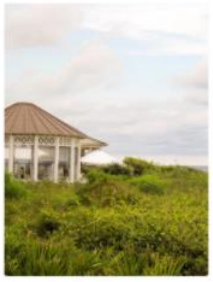
Electronics (39)

storefront_homepage_after_product_categories

storefront_homepage_before_recent_products

Recent Products

storefront_homepage_after_recent_products_title



Build Your DSLR

Read More

Lowepro Slingshot Edge 250 AW

£109.95

Add to cart

Lowepro Pro Roller x300 AW

£499.95

Add to cart

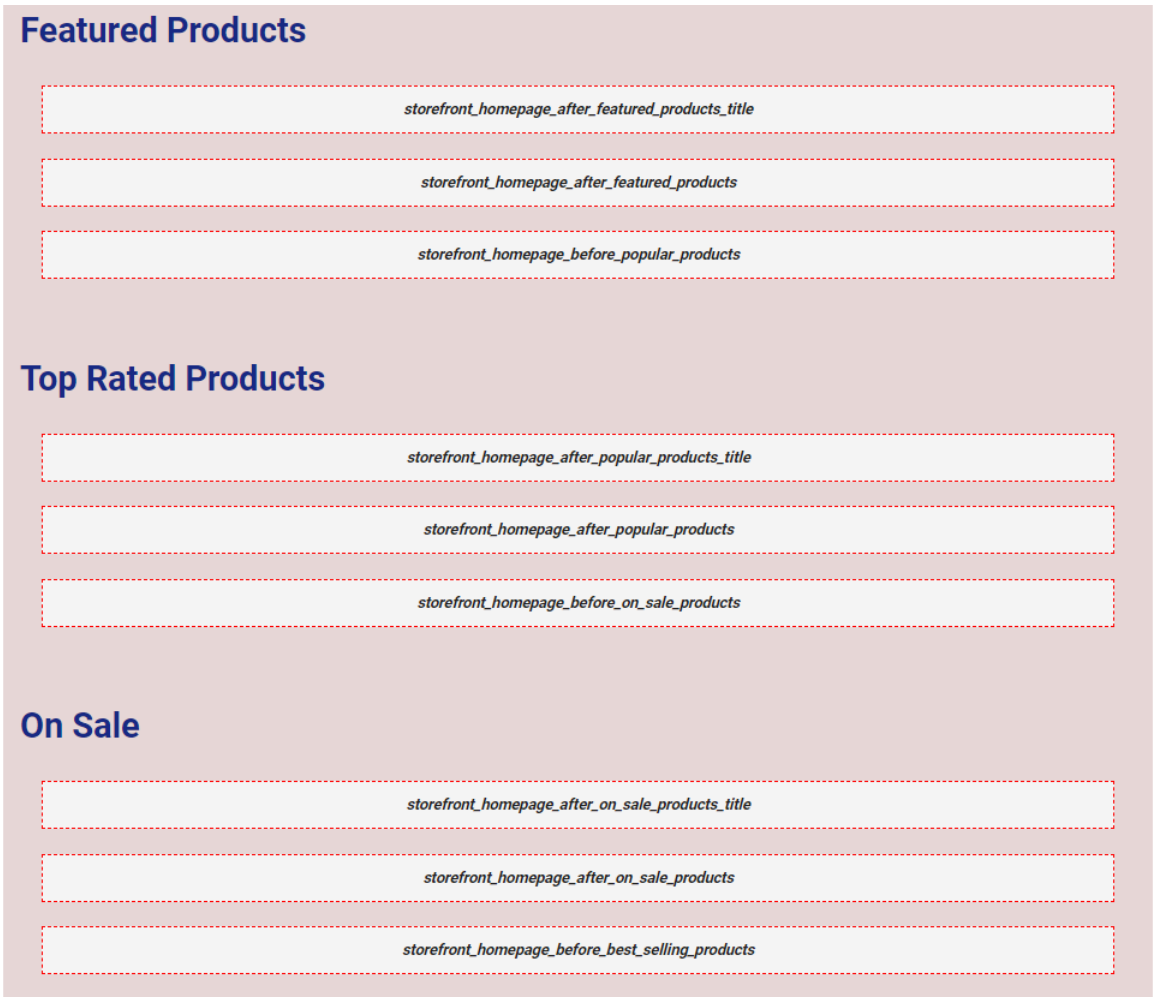
Lowepro ProTactic 350 AW

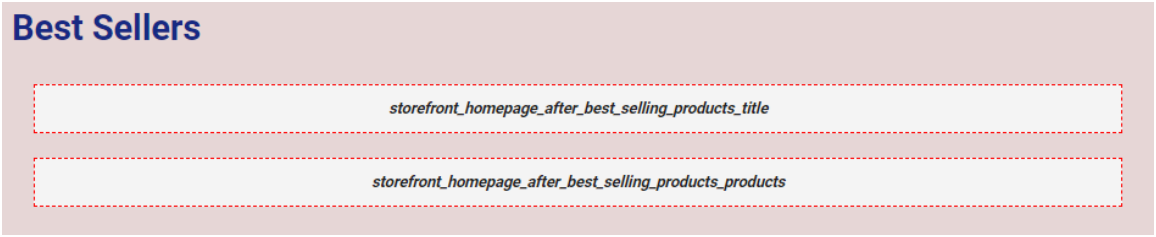
£199.95

Add to cart

storefront_homepage_after_recent_products

storefront_homepage_before_featured_products





6. Storefront Theme > Post Hooks

```
add_action( 'storefront_loop_post', 'storefront_post_header', 10 );
add_action( 'storefront_loop_post', 'storefront_post_meta', 20 );
add_action( 'storefront_loop_post', 'storefront_post_content', 30 );
add_action( 'storefront_loop_after', 'storefront_paging_nav', 10 );
add_action( 'storefront_single_post', 'storefront_post_header', 10 );
add_action( 'storefront_single_post', 'storefront_post_meta', 20 );
add_action( 'storefront_single_post', 'storefront_post_content', 30 );
add_action( 'storefront_single_post_bottom', 'storefront_post_nav', 10 );
add_action( 'storefront_single_post_bottom', 'storefront_display_comments', 20 );
add_action( 'storefront_post_content_before', 'storefront_post_thumbnail', 10 );
```

Section 6 - Hooks

https://developer.wordpress.org/plugins/hooks/

1. Introduction

To interact / modify another some code at specific, pre-defined spots (See section 4).

As foundation for how plugins and themes interact with WordPress Core.

2. Types of Hooks

1. Actions

Allow us to add data or change how WordPress operates.

Callback functions for Actions will run at a specific point in the execution of WordPress, perform some tasks, echo output (print). **Actions do not return anything back to the calling hook.**

2. Filters

Give us the ability to change or data during the execution of WordPress.

Callback functions for filters will accept a variable, modify it, and return it. It should never have side efects such as affecting global variables and output.

3. Actions vs Filters

1. Actions takes the info it receives, will do something with this information, and return nothing

2. Filters takes the info it receives, modifies something with this information (somehow), and return it

Actions Ref : https://codex.wordpress.org/Plugin_API/Action_Reference

Filter Ref : https://codex.wordpress.org/Plugin_API/Filter_Reference

Day 4 - Friday, 4th December 2020
The Coding

Section 7 - Custom Post Type

https://developer.wordpress.org/plugins/metadata/

1. Introduction

Metadata is information about information. In the case of WordPress, its information associated with posts, users, comments and terms.

2. Managing Post Metadata

1. Adding Metadata

- add_post_meta() to add metadata . The function accepts a post_id, a meta_key, a meta_value, and a unique flag.
- We may use meta_key is to make a reference between plugin and the meta value. wporg_featured_menu might be a good one. It should be noted that the same meta_key may be used multiple times to store variations of the metadata (see the unique flag below).
- The meta_value can be a string, integer, or an array. If it is an array, it will be automatically serialized before being stored in the database.
- We may use unique flag to declare whether this key should be unique. A non-unique key is something a post can have multiple variations of, like price.
- If you only ever want one price for a post, you should flag it unique and the meta_key will have one value only.

2. Updating Metadata

- update_post_meta to update the key which have already exists (accepts a post_id, a meta_key, a meta_value, and a unique flag). *Note: If you use this function and the key does NOT exist, then it will create it, as if you had used add_post_meta().

3. Deleting Metadata

- delete_post_meta() takes a post_id, a meta_key, and optionally meta_value. It does exactly what the name suggests.

4. Character Escaping

Post meta values are passed through the stripslashes() function upon being stored, so you will need to be careful when passing in values (such as JSON) that might include \ escaped characters. Consider the JSON value {"key":"value with \"escaped quotes\""}:

```php
$escaped_json = '{"key":"value with \"escaped quotes\""}';
update_post_meta( $id, 'escaped_json', $escaped_json );
$broken = get_post_meta( $id, 'escaped_json', true );
/*
$broken, after stripslashes(), ends up unparsable:
{"key":"value with "escaped quotes""}
*/
```

5. Workaround (Not sure ?)

By adding one more level of \ escaping using the function wp_slash(), you can compensate for the call to stripslashes():

```php
$escaped_json = '{"key":"value with \"escaped quotes\""}';
update_post_meta( $id, 'double_escaped_json', wp_slash( $escaped_json ) );
$fixed = get_post_meta( $id, 'double_escaped_json', true );
/*
$fixed, after stripslashes(), ends up as desired:
{"key":"value with \"escaped quotes\""}
*/
```

6. Hidden Custom Fields

- If we want to make custom fields to store parameters, (WordPress will not show custom fields which have meta_key starting with an "_" (underscore) in the custom fields list).
- add_meta_box() to show these custom fields in an unusual way by using the function.
- The example below will add a unique custom field with the meta_key name '_color' and the meta_value of 'red' but this custom field will not display in the post edit screen:

```php
add_post_meta( 68, '_color', 'red', true );
```

7. Hidden Arrays

If the meta_value is an array, it will not be displayed on the page edit screen, even if you do not prefix the meta_key name with an underscore.

3. Creating Custom Metaboxes

1. What is a Meta Box ?

- The edit screen (when we edit a post) is composed of several default boxes: Editor, Publish, Categories, Tags, etc. These boxes are meta boxes. Plugins can add custom meta boxes to an edit screen of any post type.
- The content of custom meta boxes are usually HTML form elements where the user enters data related to a Plugin's purpose, but the content can be practically any HTML you desire.

2. Why Use Meta Box ?

- Handy, flexible, modular edit screen elements that can be used to collect information related to the post being edited. A clear relationship is established because your custom meta box will be on the same screen as all the other post related information.
- Easily hidden from users that do not need to see them and displayed to those that do.
- Free to arrange the edit screen in a way that suits them, giving users control over their editing environment.

3. Adding Meta Boxes

add_meta_box() to create a meta box and plug its execution to the add_meta_boxes action hook. The following example is adding a meta box to the post edit screen and the wporg_cpt edit screen.

```
function wporg_add_custom_box() {
    $screens = [ 'post', 'wporg_cpt' ];
    foreach ( $screens as $screen ) {
        add_meta_box(
            'wporg_box_id',                 // Unique ID
            'Custom Meta Box Title',      // Box title
            'wporg_custom_box_html',  // Content callback, must be of type callabl
            $screen                            // Post type
        );
    }
}
add_action( 'add_meta_boxes', 'wporg_add_custom_box' );
```

The wporg_custom_box_html function will hold the HTML for the meta box. The following example is adding form elements, labels, and other HTML elements.

```php
function wporg_custom_box_html( $post ) {
    ?>
    <label for="wporg_field">Description for this field</label>
    <select name="wporg_field" id="wporg_field" class="postbox">
        <option value="">Select something...</option>
        <option value="something">Something</option>
        <option value="else">Else</option>
    </select>
    <?php
}
```

4. Getting Values

get_post_meta() to get the data which stored in the postmeta table. To retrieve saved user data and make use of it, you need to get it from wherever you saved it initially.

The following example enhances the previous form elements with pre-populated data based on saved meta box values. You will learn how to save meta box values in the next section.

```php
function wporg_custom_box_html( $post ) {
    $value = get_post_meta( $post->ID, '_wporg_meta_key', true );
    ?>
    <label for="wporg_field">Description for this field</label>
    <select name="wporg_field" id="wporg_field" class="postbox">
        <option value="">Select something...</option>
        <option value="something" <?php selected( $value, 'something' ); ?>>Someth
        <option value="else" <?php selected( $value, 'else' ); ?>>Else</option>
    </select>
    <?php
}
```

5. Saving Values

When a post type is saved or updated, several actions fire, any of which might be appropriate to hook into in order to save the entered values. In this example we use the save_post action hook but other hooks may be more appropriate for certain situations.

The following example will save the wporg_field field value in the _wporg_meta_key meta key, which is hidden.

```php
function wporg_save_postdata( $post_id ) {
    if ( array_key_exists( 'wporg_field', $_POST ) ) {
        update_post_meta(
            $post_id,
            '_wporg_meta_key',
            $_POST['wporg_field']
        );
    }
}
add_action( 'save_post', 'wporg_save_postdata' );
```

6. Behind the Scenes

do_meta_boxes() to display all the meta boxes that were added to the post edit screen.

7. Removing Meta Boxes

remove_meta_box() to remove an existing meta box from an edit screen. (the parameters must match with add meta box, means add_meta_box()). The default add_meta_box() calls are made from wp-includes/edit-form-advanced.php.

8. Implementation Variants

Many plugin developers find the need to implement meta boxes using various other techniques.

9. OOP

Adding meta boxes using OOP is easy and saves you from having to worry about naming collisions in the global namespace, to save memory and allow easier implementation.

```php
abstract class WPOrg_Meta_Box {


    /**
     * Set up and add the meta box.
     */
    public static function add() {
        $screens = [ 'post', 'wporg_cpt' ];
        foreach ( $screens as $screen ) {
            add_meta_box(
                'wporg_box_id',          // Unique ID
                'Custom Meta Box Title', // Box title
                [ self::class, 'html' ],   // Content callback, must be of type ca
                $screen                  // Post type
            );
        }
    }


    /**
     * Save the meta box selections.
     *
     * @param int $post_id  The post ID.
     */
    public static function save( int $post_id ) {
        if ( array_key_exists( 'wporg_field', $_POST ) ) {
            update_post_meta(
                $post_id,
                '_wporg_meta_key',
                $_POST['wporg_field']
            );
        }
    }


    /**
     * Display the meta box HTML to the user.
     *
     * @param \WP_Post $post   Post object.
     */
    public static function html( $post ) {
        $value = get_post_meta( $post->ID, '_wporg_meta_key', true );
        ?>
        <label for="wporg_field">Description for this field</label>
        <select name="wporg_field" id="wporg_field" class="postbox">
            <option value="">Select something...</option>
            <option value="something" <?php selected( $value, 'something' ); ?>>So
            <option value="else" <?php selected( $value, 'else' ); ?>>Else</option
        </select>
        <?php
    }
}

add_action( 'add_meta_boxes', [ 'WPOrg_Meta_Box', 'add' ] );
add_action( 'save_post', [ 'WPOrg_Meta_Box', 'save' ] );
```

10. AJAX

Since the HTML elements of the meta box are inside the form tags of the edit screen, the default behavior is to parse meta box values from the $_POST super global after the user have submitted the page.

- Define a Trigger #

To connect to the link, which is a change of value or any other JavaScript event.

```
/*jslint browser: true, plusplus: true */
(function ($, window, document) {
    'use strict';
    // execute when the DOM is ready
    $(document).ready(function () {
        // js 'change' event triggered on the wporg_field form field
        $('#wporg_field').on('change', function () {
            // our code
        });
    });
}(jQuery, window, document));
```

- Client-Side Code #

We need to write our client-side code, make a POST request, the response will either be success or failure, this will indicate wither the value of the wporg_field is valid.

```
/*jslint browser: true, plusplus: true */
(function ($, window, document) {
    'use strict';
    // execute when the DOM is ready
    $(document).ready(function () {
        // js 'change' event triggered on the wporg_field form field
        $('#wporg_field').on('change', function () {
            // jQuery post method, a shorthand for $.ajax with POST
            $.post(wporg_meta_box_obj.url,                          // or ajaxurl
                   {
                       action: 'wporg_ajax_change',                 // POST data, ac
                       wporg_field_value: $('#wporg_field').val() // POST data, wp
                   }, function (data) {
                        // handle response data
                        if (data === 'success') {
                            // perform our success logic
                        } else if (data === 'failure') {
                            // perform our failure logic
                        } else {
                            // do nothing
                        }
                    }
            );
        });
    });
}(jQuery, window, document));
```

- Enqueue Client-Side Code #

Put the code into a script file and enqueue it on our edit screens. In the example below we will add the AJAX functionality to the edit screens of the following post types: post, wporg_cpt. The script file will reside at /plugin-name/admin/meta-boxes/js/admin.js, plugin-name being the main plugin folder, /plugin-name/plugin.php the file calling the function.

```php
function wporg_meta_box_scripts()
{
    // get current admin screen, or null
    $screen = get_current_screen();
    // verify admin screen object
    if (is_object($screen)) {
        // enqueue only for specific post types
        if (in_array($screen->post_type, ['post', 'wporg_cpt'])) {
            // enqueue script
            wp_enqueue_script('wporg_meta_box_script', plugin_dir_url(__FILE__) .
            // localize script, create a custom js object
            wp_localize_script(
                'wporg_meta_box_script',
                'wporg_meta_box_obj',
                [
                    'url' => admin_url('admin-ajax.php'),
                ]
            );
        }
    }
}
add_action('admin_enqueue_scripts', 'wporg_meta_box_scripts');
```

- Server-Side Code #

Write our server-side code that is going to handle the request.

```php
function wporg_meta_box_scripts() {
     // Get current admin screen, or null.
    $screen = get_current_screen();
    // Verify admin screen object before use.
    if ( is_object( $screen ) ) {
        // Enqueue only for specific post types.
        if ( in_array( $screen->post_type, [ 'post', 'wporg_cpt' ], true ) ) {
            wp_enqueue_script( 'wporg_meta_box_script', plugin_dir_url( __FILE__ )
            wp_localize_script(
                'wporg_meta_box_script',
                'wporg_meta_box_obj',
                [
                    'url' => admin_url( 'admin-ajax.php' ),
                ]
            );
        }
    }
}
add_action( 'admin_enqueue_scripts', 'wporg_meta_box_scripts' );
```

4. Rendering Post Metadata

1. the_meta() - template tag that automatically lists all Custom Fields of a post.
2. get_post_custom() and get_post_meta() - get one or all metadata of a post
3. get_post_custom_values() - get values for a custom post field

Section 8 - Creating custom post types, Creating custom taxonomies, Creating custom meta boxes

https://www.smashingmagazine.com/2012/11/complete-guide-custom-post-types/

1. Create Custom Post Types (Write it in functions.php)

```
function my_custom_post_product() {
  $args = array();
  register_post_type( 'product', $args );
}
add_action( 'init', 'my_custom_post_product' );
```

To tailor our new post type to our needs, I will go through some of the more frequently used options and add them to the previously empty $args array.

```
function my_custom_post_product() {
  $labels = array(
    'name'               => _x( 'Products', 'post type general name' ),
    'singular_name'      => _x( 'Product', 'post type singular name' ),
    'add_new'            => _x( 'Add New', 'book' ),
    'add_new_item'       => __( 'Add New Product' ),
    'edit_item'          => __( 'Edit Product' ),
    'new_item'           => __( 'New Product' ),
    'all_items'          => __( 'All Products' ),
    'view_item'          => __( 'View Product' ),
    'search_items'       => __( 'Search Products' ),
```

```
    'not_found'          => __( 'No products found' ),
    'not_found_in_trash' => __( 'No products found in the Trash' ),
    'parent_item_colon'  => ’,
    'menu_name'          => 'Products'
  );

  $args = array(
    'labels'        => $labels,
    'description'   => 'Holds our products and product specific data',
    'public'        => true,
    'menu_position' => 5,
    'supports'      => array( 'title', 'editor', 'thumbnail', 'excerpt', 'comments' ),
    'has_archive'   => true,
  );
  register_post_type( 'product', $args );
}
add_action( 'init', 'my_custom_post_product' );
```

*Note:

- labels. Should be an array defining the different labels that a custom post type can have.
- description is a short explanation of our custom post type; what it does and why we are using it.
- public controls a bunch of things in one go. True will set a bunch of other options (all to do with visibility) to true. For example, it is possible to have the custom post type visible but not queryable.
- menu_position fefines the position of the custom post type menu in the back end. Setting it to “5” places it below the “posts” menu; the higher you set it, the lower the menu will be placed.
- supports set up the default WordPress controls that are available in the edit screen for the custom post type. If you want to add support for comments, revisions, post formats and such you will need to specify them here.

- has_archive if set to true, rewrite rules will be created for you, enabling a post type archive at http://mysite.com/posttype/ (by default)

2. Custom Interaction Messages

WordPress can generate a number of messages triggered by user actions, such as update, publish, search, etc. We may change this text easily by using post_updated_messages hook.

```
function my_updated_messages( $messages ) {
  global $post, $post_ID;
  $messages['product'] = array(
    0 => ’,
    1 => sprintf( __('Product updated. <a href="%s">View product</a>'), esc_url( get_permalink($post_ID) ) ),
    2 => __('Custom field updated.'),
    3 => __('Custom field deleted.'),
    4 => __('Product updated.'),
    5 => isset($_GET['revision']) ? sprintf( __('Product restored to revision from %s'), wp_post_revision_title( (int) $_GET['revision'], false ) ) : false,
    6 => sprintf( __('Product published. <a href="%s">View product</a>'), esc_url( get_permalink($post_ID) ) ),
    7 => __('Product saved.'),
    8 => sprintf( __('Product submitted. <a target="_blank" href="%s">Preview product</a>'), esc_url( add_query_arg( 'preview', 'true', get_permalink($post_ID) ) ) ),
    9 => sprintf( __('Product scheduled for: <strong>%1$s</strong>. <a target="_blank" href="%2$s">Preview product</a>'), date_i18n( __( 'M j, Y @ G:i' ), strtotime( $post->post_date ) ), esc_url( get_permalink($post_ID) ) ),
    10 => sprintf( __('Product draft updated. <a target="_blank" href="%s">Preview product</a>'), esc_url( add_query_arg( 'preview', 'true', get_permalink($post_ID) ) ) ),
  );
  return $messages;
}
add_filter( 'post_updated_messages', 'my_updated_messages' );
```

An associative array would be far better; we could see what each message is for without having to read the actual message, better to create a function for each post type just so we can group the post type creation and the custom messages together easily.

3. Create Contextual Help

The contextual help feature is a descending tab which can be seen in the top right of pages where available.

```
function my_contextual_help( $contextual_help, $screen_id, $screen ) {
  if ( 'product' == $screen->id ) {

    $contextual_help = '<h2>Products</h2>
    <p>Products show the details of the items that we sell on the website. You can see a list of them on this page in reverse chronological order - the latest one we added is first.</p>
    <p>You can view/edit the details of each product by clicking on its name, or you can perform bulk actions using the dropdown menu and selecting multiple items.</p>';

  } elseif ( 'edit-product' == $screen->id ) {

    $contextual_help = '<h2>Editing products</h2>
    <p>This page allows you to view/modify product details. Please make sure to fill out the available boxes with the appropriate details (product image, price, brand) and <strong>not</strong> add these details to the product description.</p>';

  }
  return $contextual_help;
}
add_action( 'contextual_help', 'my_contextual_help', 10, 3 );
```

Note: This is also a bit difficult because you have to know the ID of the screen you are on. If you print out the contents of the $screen variable, you should be able to determine the ID easily.

OVERVIEW

register_post_type() to create the post type itself and two hooks

contextual_help and post_updated_message to create helpful guidance and relevant messages respectively

4. Create Custom Taxonomies

Your blog posts could be about your “Life,” your “Thoughts” or your “Dreams.” These are obviously not appropriate for products. You can create a separate taxonomy named “Product Categories” to house categories you only use for products.

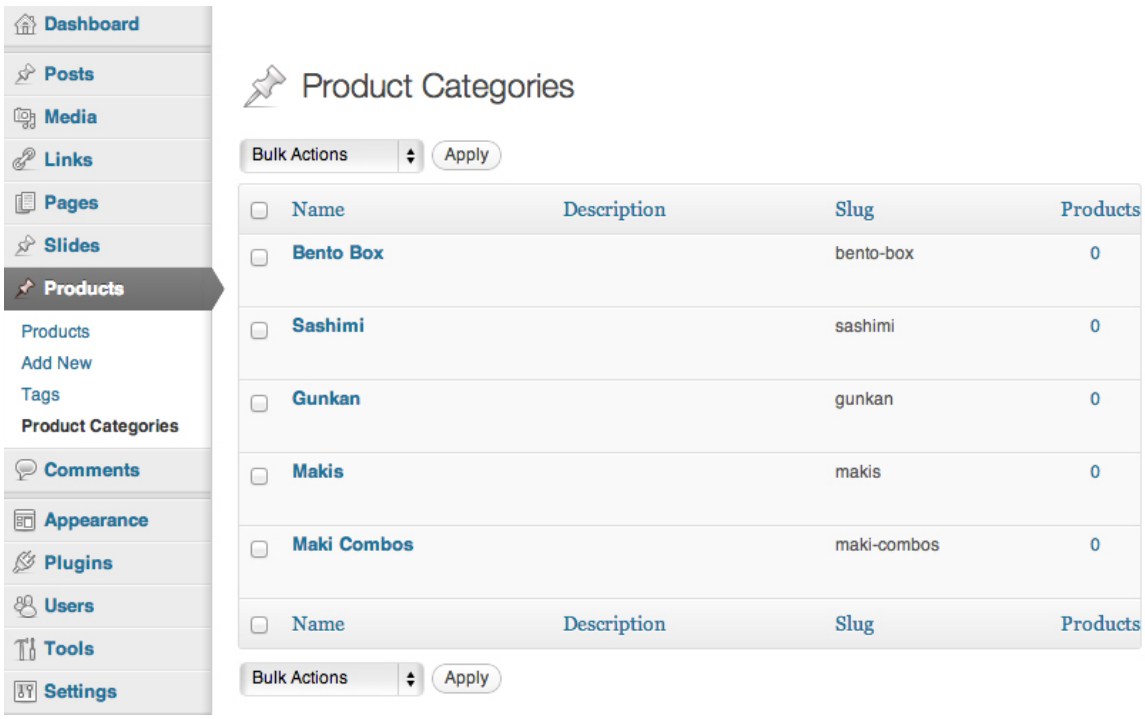
```
function my_taxonomies_product() {
  $args = array();
  register_taxonomy( 'product_category', 'product' $args );
}
add_action( 'init', 'my_taxonomies_product', 0 );
```

Custom taxonomies behave a bit better out of the box as they are public by default, so the above is actually enough to tie this taxonomy to the product posts. Let us look at a customized example.

```
function my_taxonomies_product() {
  $labels = array(
    'name'              => _x( 'Product Categories', 'taxonomy general name' ),
    'singular_name'     => _x( 'Product Category', 'taxonomy singular name' ),
    'search_items'      => __( 'Search Product Categories' ),
    'all_items'         => __( 'All Product Categories' ),
    'parent_item'       => __( 'Parent Product Category' ),
    'parent_item_colon' => __( 'Parent Product Category:' ),
    'edit_item'         => __( 'Edit Product Category' ),
    'update_item'       => __( 'Update Product Category' ),
    'add_new_item'      => __( 'Add New Product Category' ),
```

```
    'new_item_name'     => __( 'New Product Category' ),
    'menu_name'         => __( 'Product Categories' ),
  );
  $args = array(
    'labels' => $labels,
    'hierarchical' => true,
  );
  register_taxonomy( 'product_category', 'product', $args );
}
add_action( 'init', 'my_taxonomies_product', 0 );
```

We added some labels and set the hierarchical option to true. This enables “category style” taxonomies.



5. Post Meta Boxes

Meta boxes are the draggable boxes you see in the WordPress edit screen for a post. Meta boxes tend to be used to manage custom field data in a much more user-friendly way than the built-in custom fields box does.

Creating a meta box requires three steps:

1. Define The Meta Box

```
add_action( 'add_meta_boxes', 'product_price_box' );
function product_price_box() {
    add_meta_box(
        'product_price_box',
        __( 'Product Price', 'myplugin_textdomain' ),
        'product_price_box_content',
        'product',
        'side',
        'high'
    );
}
```

Parameter (In order):

- The unique identifier for the meta box (it does not have to match the function name),
- The title of the meta box (visible to users),
- The function which will display the contents of the box,
- The post type the meta box belongs to,
- The placement of the meta box,
- The priority of the meta box (determines “how high” it is placed).

2. Define The Content Of The Meta Box

```
function product_price_box_content( $post ) {
  wp_nonce_field( plugin_basename( __FILE__ ), 'product_price_box_content_nonce' );
  echo '<label for="product_price"></label>';
  echo '<input type="text" id="product_price" name="product_price" placeholder="enter a price"
/>';
}
```

This is a simple box which only contains the price, so we have created a label and an input to manage it.

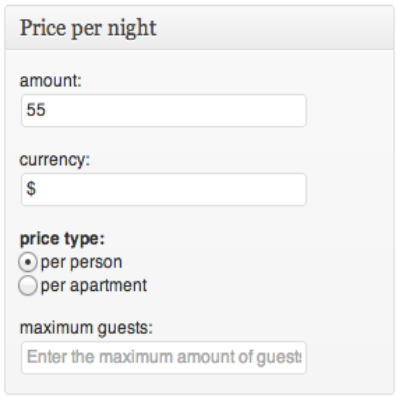
3. Handling Submitted Data

We will want to save the data as a custom field, but you are by no means restricted to this method. We could use the input to make a third-party API call, to generate an XML file or whatever you like.

```
add_action( 'save_post', 'product_price_box_save' );
function product_price_box_save( $post_id ) {

  if ( defined( 'DOING_AUTOSAVE' ) && DOING_AUTOSAVE )
  return;

  if ( !wp_verify_nonce( $_POST['product_price_box_content_nonce'],
plugin_basename( __FILE__ ) ) )
  return;

  if ( 'page' == $_POST['post_type'] ) {
    if ( !current_user_can( 'edit_page', $post_id ) )
    return;
  } else {
```

```
  if ( !current_user_can( 'edit_post', $post_id ) )
  return;
 }
 $product_price = $_POST['product_price'];
 update_post_meta( $post_id, 'product_price', $product_price );
}
```

First of all, if an autosave is being performed, nothing happens because the user has not actually submitted the form. If these are all passed, we take our data and add it to the post using the update_post_meta() function.



6. Displaying Our Content

- This involves showing posts — perhaps from various custom post types and taxonomies — and using our post metadata.

1. Displaying Posts

If you have created a post type with the has_archive parameter set to “true,” WordPress will list your posts on the post type’s archive page. If your post type is called “books,” you can simply go to http://mysite.com/books/ and you will see your post list.

Note:

- This page uses archive-[post_type].php for the display if it exists (archive-books.php in our case). If it does not exist, it will use archive.php and if that does not exist it will use index.php.
- Another way to display custom post type content is to use a custom query with the WP_Query class.

```
<?php
  $args = array(
   'post_type' => 'product',
   'tax_query' => array(
    array(
     'taxonomy' => 'product_category',
     'field' => 'slug',
     'terms' => 'boardgames'
    )
   )
  );
  $products = new WP_Query( $args );
  if( $products->have_posts() ) {
   while( $products->have_posts() ) {
    $products->the_post();
    ?>
     <h1><?php the_title() ?></h1>
     <div class='content'>
      <?php the_content() ?>
     </div>
    <?php
   }
  }
  else {
   echo 'Oh ohm no products!';
```

```
  }
?>
```

2. Displaying Meta Data

get_post_meta() to retrieve metadata easily. In our example above, we saved a post meta field named product_price. We can retrieve the value of this field for a given post using the following code:

```
<?php
  // If we are in a loop, we can get the post ID easily
  $price = get_post_meta( get_the_ID(), 'product_price', true );

  // To get the price of a random product we will need to know the ID
  $price = get_post_meta( $product_id, 'product_price', true );
?>
```

Section 10 - Wordpress Prebuilt Functions

https://vegibit.com/the-top-100-most-commonly-used-wordpress-functions/

1. get_theme_mod()
Retrieve theme modification value for the current theme.

2. add_setting()
Add a customize setting.

4. apply_filters()
Call the functions added to a filter hook.

5. esc_attr()
Escaping for HTML attributes.

6. get_option()
Retrieves an option value based on an option name.

7. esc_url()
Checks and cleans a URL.

8. esc_html()
Escaping for HTML blocks.

9. _e()

In WordPress, strings in the php files are marked for translation to other languages, and localization using two functions: __() and _e().

10. absint()
Convert a value to non-negative integer.

11. get_template_part()
Loads a template part into a template.

12. is_singular()
Is the query for an existing single post of any post type (post, attachment, page, custom post types)?

13. get_post_type()
Retrieves the post type of the current post or of a given post.

14. get_the_ID()
Retrieve the ID of the current item in the WordPress Loop.

15. the_content()
Display the post content.

16. have_posts()
Whether current WordPress query has results to loop over.

17. post_class()
Display the classes for the post div.

18. get_comments_number()
Retrieves the number of comments a post has.

19. the_ID()
Display the ID of the current item in the WordPress Loop.

20. is_single()
Is the query for an existing single post?

21. get_permalink()
Retrieves the full permalink for the current post or post ID.

22. get_the_title()
Retrieve post title.

23. admin_url()
Retrieves the URL to the admin area for the current site.

24. add_section()
Add a customize section.

25. is_home()
Determines if the query is for the blog homepage.

26. get_sidebar()
Load sidebar template.

27. get_footer()
Load footer template.

28. is_customize_preview()
Whether the site is being previewed in the Customizer.

29. wp_nav_menu()
Displays a navigation menu.

30. the_title()
Display or retrieve the current post title with optional markup.

31. current_user_can()
Whether the current user has a specific capability.

32. is_active_sidebar()
Whether a sidebar is in use.

33. wp_link_pages()
The formatted output of a list of pages.

34. get_the_time()
Retrieve the time at which the post was written.

35. get_the_date()
Retrieve the date on which the post was written.

36. the_permalink()
Displays the permalink for the current post.

37. the_post()
Iterate the post index in the loop.

38. esc_attr_e()
Display translated text that has been escaped for safe use in an attribute.

39. is_front_page()
Is the query for the front page of the site?

40. bloginfo()
Displays information about the current site.

41. comments_open()
Whether the current post is open for comments.

42. post_password_required()
Whether post requires password and correct password has been provided.

43. has_post_thumbnail()
Check if post has an image attached.

44. get_bloginfo()
Retrieves information about the current site.

45. get_post_format()
Retrieve the format slug for a post

46. dynamic_sidebar()
Display dynamic sidebar.

47. is_search()
Is the query for a search?

48. home_url()
Retrieves the URL for the current site where the front end is accessible.

49. comments_template()
Load the comment template specified in $file.

50. add_theme_support()
Registers theme support for a given feature.

51. add_query_arg()
Retrieves a modified URL query string.

52. has_nav_menu()
Determines whether a registered nav menu location has a menu assigned to it.

53. is_wp_error()
Check whether variable is a WordPress Error.

54. get_the_author_meta()
Retrieves the requested data of the author of the current post.

55. edit_post_link()
Displays the edit post link for post.

56. is_admin()
Whether the current request is for an administrative interface page.

57. the_excerpt()
Display the post excerpt.

58. wp_get_attachment_image_src()
Retrieve an image to represent an attachment.

59. _x()
Retrieve translated string with gettext context.

60. language_attributes()
Displays the language attributes for the html tag.

61. body_class()
Display the classes for the body element.

62. add_filter()
Hook a function or method to a specific filter action.

63. is_page()
Is the query for an existing single page?

64. register_sidebar()
Builds the definition for a single sidebar and returns the ID.

65. get_the_category_list()
Retrieve category list for a post in either HTML list or custom format.

66. get_the_tag_list()
Retrieve the tags for a post formatted as a string.

67. esc_attr_x()
Translate string with gettext context and escapes it for safe use in an attribute.

68. get_setting()
Get value based on option. (Use get_option() instead.)

69. add_action()
Hooks a function on to a specific action.

70. have_comments()
Whether there are comments to loop over.

71. is_archive()
Is the query for an existing archive page?

72. wp_list_comments()
List comments.

73. wp_enqueue_style()
Enqueue a CSS stylesheet.

74. set_transient()
Set/update the value of a transient.

75. wp_enqueue_script()
Enqueue a script.

76. get_search_form()
Display search form.

77. get_post_thumbnail_id()
Retrieve post thumbnail ID.

78. get_transient()
Get the value of a transient.

79. the_post_thumbnail()
Display the post thumbnail.

80. get_search_query()
Retrieves the contents of the search WordPress query variable.

81. add_partial()
add_partial()

82. get_the_modified_date()
Retrieve the date on which the post was last modified.

83. get_author_posts_url()
Retrieve the URL to the author page for the user with the ID provided.

84. wp_footer()
Fire the wp_footer action.

85. wp_head()
Fire the wp_head action.

86. get_the_post_thumbnail()
Retrieve the post thumbnail.

87. comment_form()
Outputs a complete commenting form for use within a template.

88. number_format_i18n()
Convert float number to format based on the locale.

89. get_header()
Load header template.

90. the_posts_pagination()
Displays a paginated navigation to next/previous set of posts, when applicable.

91. register_nav_menus()
Registers navigation menu locations for a theme.

92. wp_die()
Kill WordPress execution and display HTML message with error message.

93. wp_reset_postdata()
After looping through a separate query, this function restores the $post global to the current post in the main query.

94. load_theme_textdomain()
Load the theme's translated strings.

95. get_queried_object_id()
Retrieve ID of the current queried object.

96. esc_url_raw()
Performs esc_url() for database usage.

97. the_archive_title()
Display the archive title based on the queried object.

98. add_image_size()
Register a new image size.

99. get_theme_file_uri()
Retrieves the URL of a file in the theme.

100. get_stylesheet_uri() Retrieves the URI of current theme stylesheet.

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

Some functions needed:

- activate_plugin()

Function: Attempts activation of plugin in a “sandbox” and redirects on success.

- activate_plugins()

Function: Activate multiple plugins.

- add_action()

Function: Hooks a function on to a specific action.

- add_filter()

Function: Hook a function or method to a specific filter action.

- add_meta()

Function: Add post meta data defined in $_POST superglobal for post with given ID.

- add_metadata()

Function: Adds metadata for the specified object.

- add_meta_box()

Function: Adds a meta box to one or more screens.

- add_posts_page()

Function: Add submenu page to the Posts main menu.

- add_role()

Function: Add role if it does not exist.

- add_shortcode()

Function: Adds a new shortcode.

- add_user()

Function: Creates a new user from the “Users” form using $_POST information.

- allowed_tags()

Function: Display all of the allowed tags in HTML format with attributes.

- current_time()

Function: Retrieves the current time based on specified type.

- delete_meta()

Function: Delete post meta data by meta-ID.

- doing_action()

Function: Retrieve the name of an action currently being processed.

- doing_filter()

Function: Retrieve the name of a filter currently being processed.

- do_shortcode()

Function: Search content for shortcodes and filter shortcodes through their hooks.

Section 11 - Hooks by Han

https://www.youtube.com/watch?v=9GuJi8dYuAs

1. Type of Hooks

1. Actions which allow us to run code when an event happens in WordPress

2. Filters which allow us to modify data before it is either sent to the database displayed in the browser.

2. Write functions.php

```
add_action('save_post', 'log_when_saved');
function log_when_saved( $post_id ) {

 $post_log = get_stylesheet_directory() . '/post_log.txt';
 $message = get_the_title( $post_id ) . ' was just saved!';

 if ( file_exists( $post_log ) ) {

   $file = fopen( $post_log, 'a' );
   fwrite($file, $message."\n");

 } else {

   $file = fopen( $post_log, 'w' );
   fwrite( $file, $message."\n" );

 }
```

```
 fclose($file);

}
```

Note:

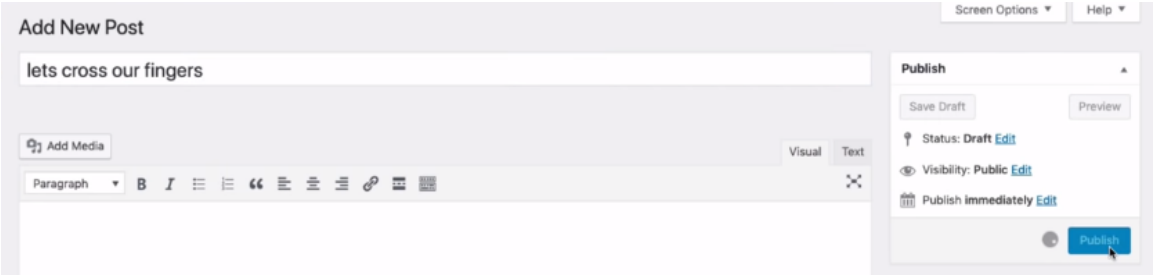
1. add_action() is when we need run code when something else is happening in WordPress.

add_action('p1', 'p2').

p1 = the name of the action that we want to hook into

p2 = the function that you want to run when the first parameter happens

2. Add a post in WordPress, Publish.



This post will print lest cross our fingers two times because this posts run not only when we save (or publish) a post, but also when a post has been revised and when draft is created. Therefore, you need to modify it.

```
add_action('save_post', 'log_when_saved');
function log_when_saved( $post_id ) {

 if( ! ( wp_is_post_revision( $post_id ) ) || wp_is_post_autosave( $post_id ) ) {
   return;
 }

 $post_log = get_stylesheet_directory() . '/post_log.txt';
 $message = get_the_title( $post_id ) . ' was just saved!';

 if ( file_exists( $post_log ) ) {
```

3. Scenario 2:

We have a page on our site and called super-secret and we dont want anybody who is not logged in to WordPress to see this page. Therefore, we use template_redirect.

```
add_action('template_redirect', 'members_only');
function members_only(){

 if( is_page('super-secret') && ! is_user_logged_in() ) {
   wp_redirect( home_url() );
   die();
 }

}
```

4. Scenario 3: write the log if not user access to our secret page.

```
add_action('template_redirect', 'members_only');
function members_only(){

 if( is_page('super-secret') && ! is_user_logged_in() ) {
   do_action('user_redirected', date("F j, Y, g:i a"));
   wp_redirect( home_url() );
   die();
 }

}
```

```
add_action('user_redirected', 'log_when_accessed');
function log_when_accessed( $date ) {

$access_log = get_stylesheet_directory() . '/access_log.txt';
$message = 'someone just tried to access our super secret page on ' . $date;

if ( file_exists( $access_log ) ) {

  $file = fopen( $access_log, 'a' );
  fwrite($file, $message."\n");

} else {

  $file = fopen( $access_log, 'w' );
  fwrite( $file, $message."\n" );

}

fclose($file);
```

10:35 / 12:10

do_action() is when we create our own custom actions.

Section 5 - Debug Logs

Part I

https://www.wpwhitesecurity.com/complete-guide-wordpress-debug-mode/

Sometimes we are unable to reproduce a problem that a user is encountering when using one of our WordPress plugins. When this happens, we ask the user to send us more information about their website's and plugin's setup.

What is the purpose of WordPress debug?

In case of present issues on your website, the WordPress debug mode can help you discover what is going wrong on your website.

Enabling WordPress Debugging

To enable the WordPress debug mode and the debug log file follow the below procedure:

- Enable WordPress debug by setting the WP_DEBUG switch in the wp-config.php file to true, as per the following example: define('WP_DEBUG', true);
- Enable the debug log file by adding the following line to the wp-config.php file: define('WP_DEBUG_LOG', true);

The WordPress debug log file is called debug.log and it is created in the /wp-content/ WordPress subdirectory.

Additional WordPress Debugging Options

1. WP_DEBUG_DISPLAY

When this option is enabled alongside the WordPress debug switch, WordPress shows the debug messages in the HTML pages as they are generated.

Note that if this is used on a live website this could lead to the disclosure of sensitive information about the website and server setup. To enable this option, add the below line to your wp-config.php file:

```
define( 'WP_DEBUG_DISPLAY', true );
```

2. SCRIPT_DEBUG

When this option and WordPress debugging are enabled, WordPress uses the development versions of core CSS and JavaScript files instead of the minified (compressed) version that it normally uses.

This debugging option is useful if you are testing changes in .js and .css files. To enable this option, add the below line to your wp-config.php file:

define('SCRIPT_DEBUG', true);

3. Logging of WordPress Database Queries

On WordPress, you can also keep a log of the WordPress database query in an array. This option comes in handy if you are experiencing WordPress database issues, or you want to check what queries are being sent.

To enable it, add the following line to your WordPress wp-config.php file:

define('SAVEQUERIES', true);

Once you enable this option all queries will be saved in the $wpdb->queries global.

Debugging Affect The Performance of WordPress Website

Yes. Some debugging options can have an impact on the performance of your website. Therefore, unless it is a test or staging website, or you are instructed to temporarily enable debug by developers do not enable debugging on live websites.

Part II

https://wordpress.org/support/article/debugging-in-wordpress/

1. WP_DEBUG

A PHP constant (a permanent global variable) that can be used to trigger the "debug" mode throughout WordPress.

```
// This enables debugging.
define( 'WP_DEBUG', true );
```

```
// This disables debugging.
define( 'WP_DEBUG', false );
```

PHP Errors, Warnings, and Notices

Enabling WP_DEBUG will cause all PHP errors, notices and warnings to be displayed. This is likely to modify the default behavior of PHP which only displays fatal errors and/or shows a white screen of death when errors are reached.

Deprecated Functions and Arguments

Enabling WP_DEBUG will also cause notices about deprecated functions and arguments within WordPress that are being used on your site. These are functions or function arguments that have not been removed from the core code yet but are slated for deletion in the near future.

2. WP_DEBUG_LOG

WP_DEBUG_LOG is a companion to WP_DEBUG that causes all errors to also be saved to a debug.log log file This is useful if you want to review all notices later or need to view notices generated off-screen (e.g., during an AJAX request or wp-cron run).

When set to true, the log is saved to debug.log in the content directory (usually wp-content/debug.log) within your site's filesystem. Alternatively, you can set it to a valid file path to have the file saved elsewhere.

```
define( 'WP_DEBUG_LOG', true );
-or-
define( 'WP_DEBUG_LOG', '/tmp/wp-errors.log' );
```

Note: for WP_DEBUG_LOG to do anything, WP_DEBUG must be enabled (true). Remember you can turn off WP_DEBUG_DISPLAY independently.

3. WP_DEBUG_DISPLAY

WP_DEBUG_DISPLAY is another companion to WP_DEBUG that controls whether debug messages are shown inside the HTML of pages or not. The default is 'true' which shows errors and warnings as they are generated. Setting this to false will hide all errors.

```
define( 'WP_DEBUG_DISPLAY', false );
```

Note: for WP_DEBUG_DISPLAY to do anything, WP_DEBUG must be enabled (true). Remember you can control WP_DEBUG_LOG independently.

4. SCRIPT_DEBUG

SCRIPT_DEBUG is a related constant that will force WordPress to use the "dev" versions of core CSS and JavaScript files rather than the minified versions that are normally loaded. This is useful when you are testing modifications to any built-in .js or .css files. Default is false.

```
define( 'SCRIPT_DEBUG', true );
```

5. SAVE QUERIES

The SAVEQUERIES definition saves the database queries to an array and that array can be displayed to help analyze those queries. The constant defined as true causes each query to be saved, how long that query took to execute, and what function called it.

```
define( 'SAVEQUERIES', true );
```

NOTE: This will have a performance impact on your site, so make sure to turn this off when you are not debugging.

Example wp-config.php for Debugging

The following code, inserted in your wp-config.php file, will log all errors, notices, and warnings to a file called debug.log in the wp-content directory. It will also hide the errors, so they do not interrupt page generation.

```
// Enable WP_DEBUG mode
define( 'WP_DEBUG', true );

// Enable Debug logging to the /wp-content/debug.log file
define( 'WP_DEBUG_LOG', true );

// Disable display of errors and warnings
define( 'WP_DEBUG_DISPLAY', false );
@ini_set( 'display_errors', 0 );

// Use dev versions of core JS and CSS files (only needed if you are modifying these core files)
define( 'SCRIPT_DEBUG', true );
```

NOTE: You must insert this BEFORE /* That's all, stop editing! Happy blogging. */ in the wp-config.php file.

Debugging Plugins

There are many debugging plugins for WordPress that show more information about the internals, either for a specific component or in general. Here are some examples:

- Query Monitor
- Debug Bar
- Log Deprecated Notices

Section 9 - Advanced Custome Fields

Structure Learning With Jason

folder structure => where coding

database stucture

wp method:hook、

wc 購物流程、

會開放給客戶的權限、

WP Query、

get_post、

WC()、

$wpdb、

comment format、

ManageWP、

DropBox、

主機相關(Linode)

、CF、

Godaddy、

Bitbucket、

Alan 做的雲端管理列表、

WPML、

loco translate、

帳密從 trello 找、

客戶資料檔案從 Google Drive 找、

trello 專案格式(專案主機放到 trello 的 Engineer Info)、

建測試站流程、

獨立建站流程、

如何覆寫模板、

開案或結案(結案前要做的事情有比對報價單、去掉不必要的、比對功能列表/時程表，陸續都要填寫完成的時程表項目，開案時會大家所有案子成員一起討論)

Note Jason:

Part 1

Database that i need always use comments, comments meta, users, users meta, terms, terms meta, post, poste meta, options (options are something about global settings,m for example site name, site url, admin email, etc)

Part 2

If the web vrash maybe becasue of lugins so ou may come to opstions table and change the value of active plugins, (cut) and deactive all the plugins and active it one by one to know which plugins make the cause. Maybe if our website redirect to another website maybe because of site url and home in options table

Post and Post Meta Table

The most important table in WordPress, many post types, like post, pages, (post and pages are common use), clients , offer, and slides (but these threes are not common), also for testimonials, layout, templates (8). we can add post in our php.

Posts : always display in blog

Pages : always shop the name content about home page, every single link

code:

get post : get_post_meta will get data from site_postmeta table (common to use)

get_posts to get all your posts., and we can put some query like below.

```
$r = $wpdb->gt_results("Selecet ").
foreach($r as $note) {
...
```

}

Taxonomy is like a class, we can hav emore than 2 but default tag and categories (group) Terms are taxonomy

Part 3 - Hooks

Filter and Actions

Every new project you need to disable updated theme and updated plugin (2 hooks)

Note:

- You can install simply show hooks to show all the hooks in wordpress.
- If you know the hooks name give the title of the function and put your functions below all the hooks.

Part 4 - Cart System Procedure

Using shortcode : [woocommer_cart][woocommer_checkout][woocommer_my_account] Woocommerce > Cart System is used to modify shipping destination, use tax, currency.

The pages of the woocomerce

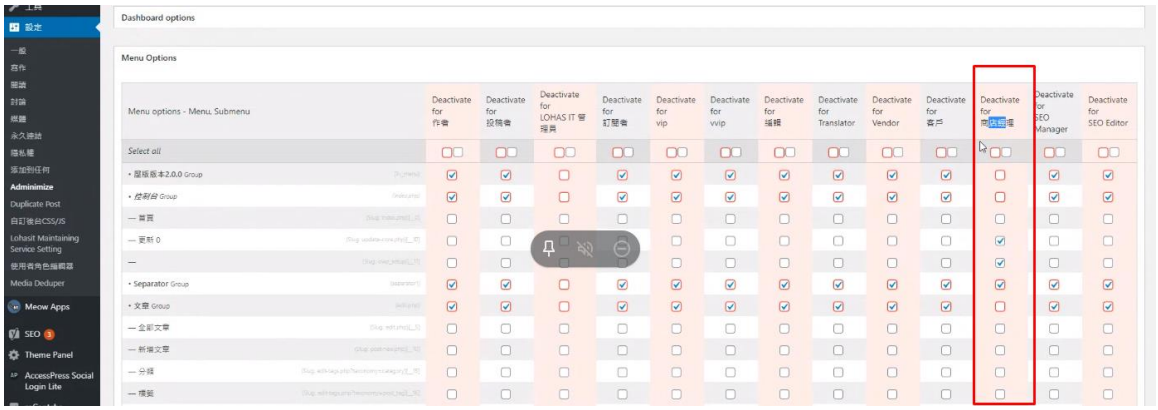
[yith_wcwl_wishlist]

If we want to mdify woocomerce for exaple cart.php so create folder woocomere, cart, cart.php skip the templates.

Part 5 - Role and Permission

1.Settings > Adminize (plugin name)

Teach the customer how to use, before this, we need to close some permission or deny some permission. Update first, so we may get the last role.



Red : Parent option

Grey : Child option

Check : Disabled

LohasIT : Our account, no need to check something

For all users you just need to disable the parent (red), so in here we just use LohasIT and Shopkeeper.

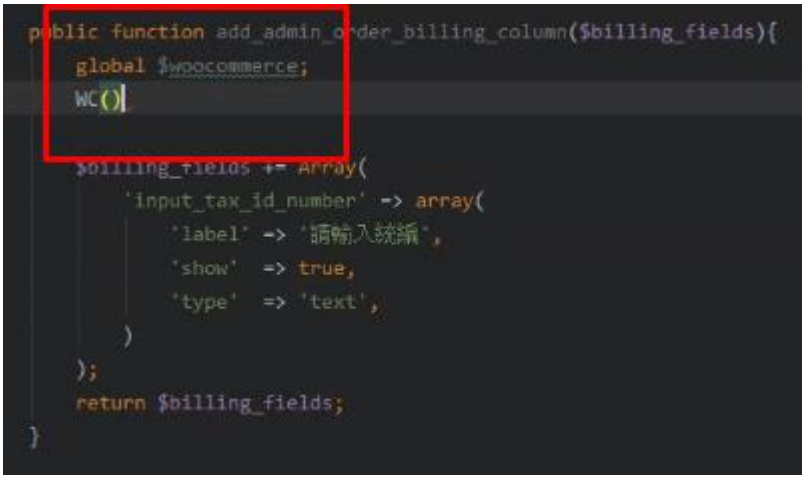
Shopkeeper function depends on the customer requirements, ask the designer or boss, you can compare with the demo website (mostly project will be like this)

2. User / Accounts > User Role Editor

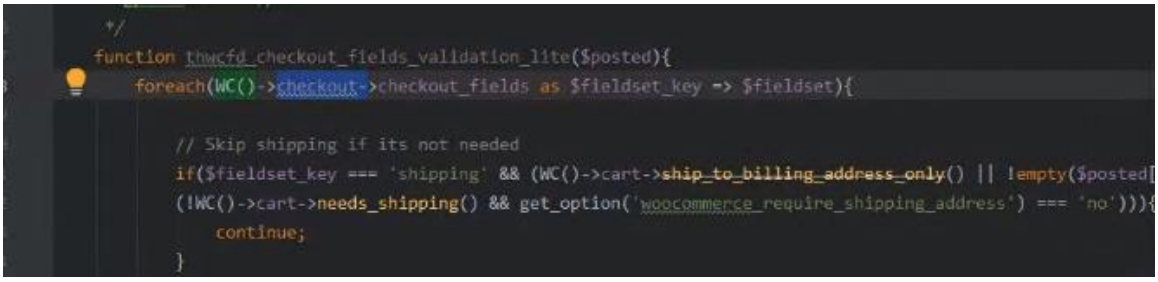
Every row has been abilities : Change shop manager and check the ability (default, 1st time whey will check there, but sometimes we need to change)

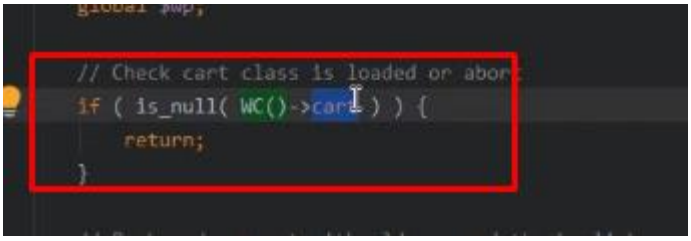
get_terms() get all the terms in databae, terms are like

Part 6 - WC()



Same Declaration





Part 7 - Comment Format

// LohasIT cart modified

Part 8 - ManageWP

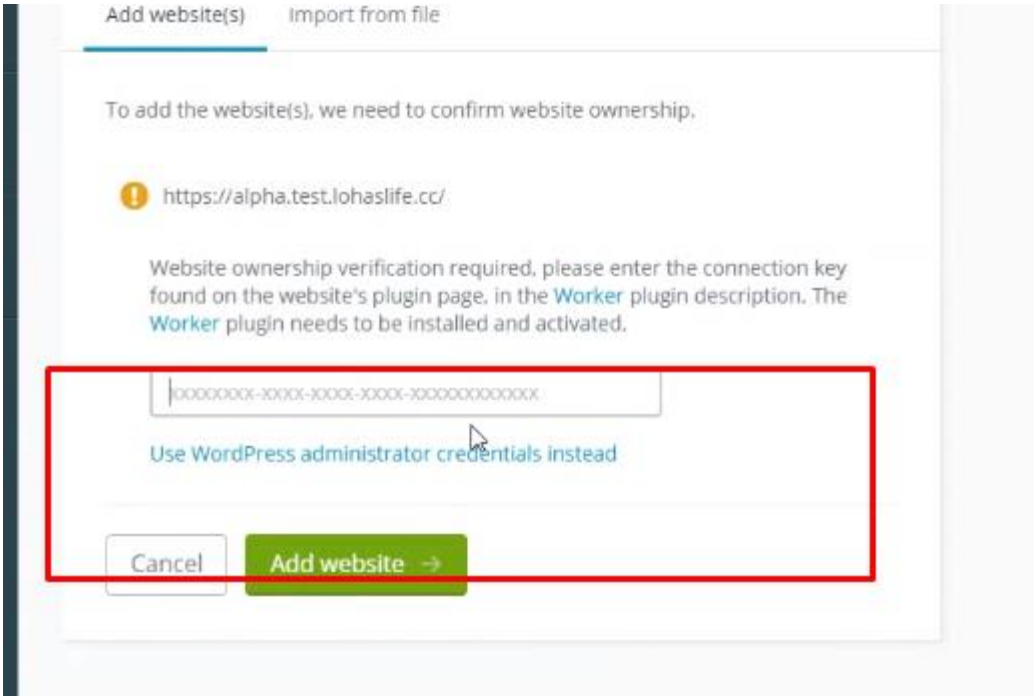
For wordpress backup (?)

https://docs.google.com/spreadsheets/d/12SC1-Gr635NB_vVyUs5EaKsoA38_wne32nXzTKU6rqw/edit#gid=0

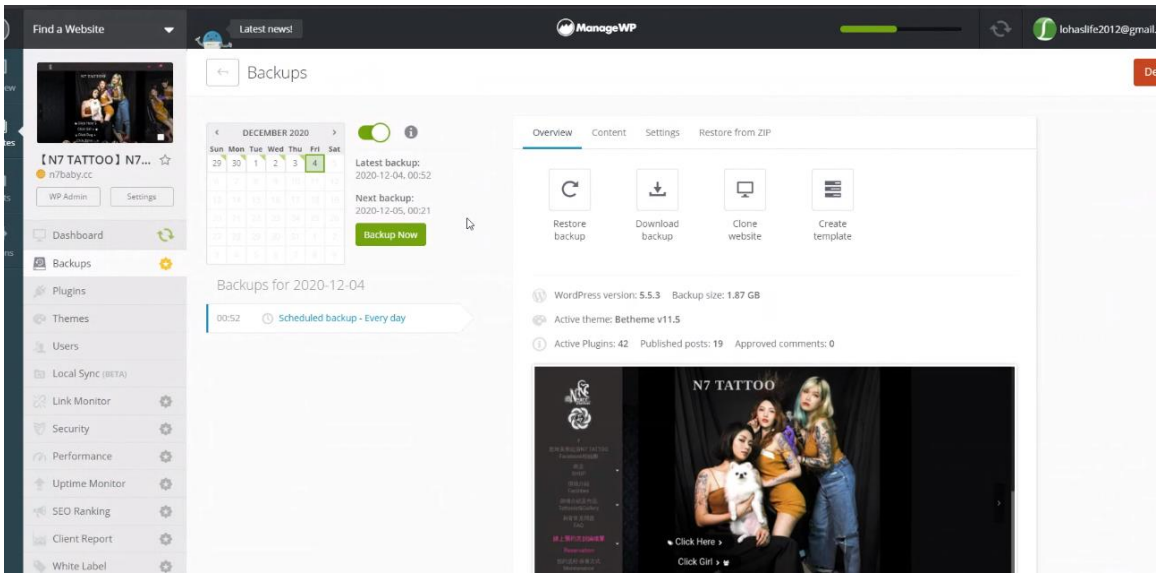
1. Add new website, Install a plugin


連線管理
這裡列出這個外掛目前進行的連線
ID
已連線
最後使用時間
5708258
10 個月前
5 個月前
中斷連線
1683378
5 個月前
2 個月前
中斷連線
*
無法使用
1 秒前
中斷連線
連線金鑰
fe06390f-43f9-433a-b96f-66a2f1ea0760
複製

and paste

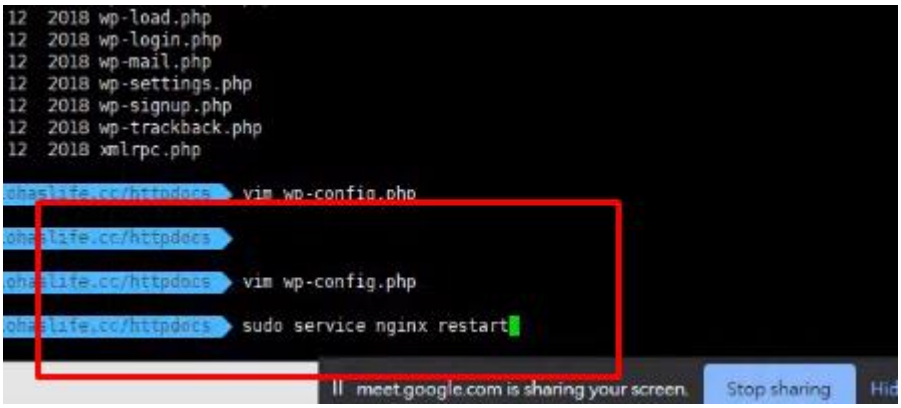

Add website(s)
Import from file
To add the website(s), we need to confirm website ownership.
https://alpha.test.lohaslife.cc/
Website ownership verification required, please enter the connection key found on the website's plugin page, in the Worker plugin description. The Worker plugin needs to be installed and activated.
xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
Use WordPress administrator credentials instead
Cancel
Add website

2. Active 1 plan



3. Click everyday plan

To restart the server



WPML and Loco language to translate