### Часть 3

## Создание темы Law

**Урок 1 - Стартовая тема и шаблон**

Подготовливаем директорию и создаем стартовую тему.

preview: <https://www.free-css.com/assets/files/free-css-templates/preview/page232/law/>

download template: <https://www.free-css.com/free-css-templates/page232/law>

Создаем домен сайта (директорию для сайта) на сервере -> Устанваливаем WP -> в папку themeы помещаем стартовую тему с underscores -> перезапускаем сервер.

Создаем БД для сайта.

Открываем law.loc в браузере -> устанивливаем WP и настравиваем config для подключения к БД.

В папке с темой можно создать временную папку template/ для хранения исходного макета сайта.

В корень темы кидаем screenshot.png чтобы в админке у темы была картинка preview.

**Урок 2 - Подключение стилей и скриптов**

Создаем assets/ в корне темы -> копируем в assets/ папки из исходного шаблона (css/, js/, fonts/ и т.д.).

В этой части автор решил пойти другим путем по переносу исходного шаблона в WP.

Поэтому берем содержимое index.html из исходного шаблона, и под чистую заменяем им содержимое файла index.php, в разрабатываемой теме.

Далее необходимо подключить скрипты и стили в functions.php:

Создаем свой файл стилей law/assets/css/law.css (чтобы наши стили не перемешивались с исходними стилями \_underscores).

В index.php смотрим необходимые скрипты для подключения и подключаем их через functions.php в law\_scripts()

*/\*\*  
 \* Enqueue scripts and styles.  
 \*/***function** law\_scripts() {  
. . . . . . . . . . . . . . . . .

. . . . . . . . . . . . . . . . .

. . . . . . . . . . . . . . . . .

. . . . . . . . . . . . . . . . .

/\*My scripts and styles  
\* header part  
\*/  
 wp\_enqueue\_style( 'law-googlefonts','https://fonts.googleapis.com/css?family=Work+Sans:300,400,500,700,800' );  
 wp\_enqueue\_style( 'law-animate', get\_template\_directory\_uri() . '/assets/css/animate.css' );  
 wp\_enqueue\_style( 'law-icomoon', get\_template\_directory\_uri() . '/assets/css/icomoon.css' );  
 wp\_enqueue\_style( 'law-bootstrap', get\_template\_directory\_uri() . '/assets/css/bootstrap.css' );  
 wp\_enqueue\_style( 'law-magnific-popup',get\_template\_directory\_uri() . '/assets/css/magnific-popup.css' );  
 wp\_enqueue\_style( 'law-owl-carousel',get\_template\_directory\_uri() . '/assets/css/owl.carousel.min.css' );  
 wp\_enqueue\_style( 'law-owl-theme-default',get\_template\_directory\_uri() . '/assets/css/owl.theme.default.min.css' );  
 wp\_enqueue\_style( 'law-flexslider',get\_template\_directory\_uri() . '/assets/css/flexslider.css' );  
 wp\_enqueue\_style( 'law-theme',get\_template\_directory\_uri() . '/assets/css/style.css' );  
 /\* my styles law.css\*/  
 wp\_enqueue\_style( 'law-custom',get\_template\_directory\_uri() . '/assets/css/law.css' );  
  
 wp\_enqueue\_script( 'law-modernizr-js', get\_template\_directory\_uri() . '/assets/js/modernizr-2.6.2.min.js' );  
 wp\_enqueue\_script( 'law-respond-js', get\_template\_directory\_uri() . '/assets/js/respond.min.js' );  
  
 /\* footer part \*/  
 /\* Подключаем Last.ver - JQuery  
 \* Подключаем именно в header так как эту библ. будет использ. слайдер из unyson  
 \*/  
 wp\_deregister\_script( 'jquery' );  
 wp\_register\_script(  
 'jquery',  
 get\_template\_directory\_uri() . '/assets/js/jquery-3.4.1.min.js'  
 );  
 wp\_enqueue\_script('jquery');  
 wp\_enqueue\_script(  
 'law-jquery-easing-js',  
 get\_template\_directory\_uri() . '/assets/js/jquery.easing.1.3.js', **array**(), '', **true** );  
 wp\_enqueue\_script(  
 'law-bootstrap-js',  
 get\_template\_directory\_uri() . '/assets/js/bootstrap.min.js', **array**(), '', **true** );  
 wp\_enqueue\_script(  
 'law-jquery-waypoints-js',  
 get\_template\_directory\_uri() . '/assets/js/jquery.waypoints.min.js', **array**(), '', **true** );  
 wp\_enqueue\_script( //этот атвор пропустил  
 'law-jquery-stellar-js',  
 get\_template\_directory\_uri() . '/assets/js/jquery.stellar.min.js', **array**(), '', **true** );  
 wp\_enqueue\_script(  
 'law-owl-carousel-js',  
 get\_template\_directory\_uri() . '/assets/js/owl.carousel.min.js', **array**(), '', **true** );  
 wp\_enqueue\_script(  
 'law-jquery-flexslider-js',  
 get\_template\_directory\_uri() . '/assets/js/jquery.flexslider-min.js', **array**(), '', **true** );  
 wp\_enqueue\_script(  
 'law-jquery-countTo-js',  
 get\_template\_directory\_uri() . '/assets/js/jquery.countTo.js', **array**(), '', **true** );  
 wp\_enqueue\_script(  
 'law-jquery-magnific-popup-js',  
 get\_template\_directory\_uri() . '/assets/js/jquery.magnific-popup.min.js', **array**(), '', **true** );  
 wp\_enqueue\_script(  
 'law-magnific-popup-options-js',  
 get\_template\_directory\_uri() . '/assets/js/magnific-popup-options.js', **array**(), '', **true** );  
 wp\_enqueue\_script(  
 'law-main',  
 get\_template\_directory\_uri() . '/assets/js/main.js', **array**(), '', **true** );

}

так как в wp\_enqueue\_style() и wp\_enqueue\_script() есть heandler(ы) с одинаковыми именами, автор присвоил heandler(ам) из wp\_enqueue\_script() постфикс -js, чтобы не было пересечения со style.

Далее необходимо полностью очистить <header></header> -> вызвать wp\_head() в <header>

Затем чистим футер, т.е. убираем явное подключение скриптов и вызываем

<?php wp\_footer(); ?> перед закрывающим тегом </body>

Теперь необходимо настроить вывод картинок:

сначала в index.php находим подстроку 'url(images', выделяем ее и наж-ем ctrl+r, затем указываем чем заменять: url(<?php echo get\_template\_directory\_uri(); ?>/assets/images  
и наж-ем replace all чтобы заменить все подобные вхождения, получится вот так:

<div class="video fh5co-video" style="background-image: url(**<?php** get\_template\_directory\_uri(); **?>**/assets/images/video.jpg);">

Дальше ищем остальные вхождения для подключения картинок с источником src, также при помощи replace all меняем все вхожения src="images на src="<?php echo get\_template\_directory\_uri(); ?>/assets/images

Автор начал создавать тему с полного переноса шаблона index.html в index.php (что не обязательно) во избежании ошибок, так как страница большая и содержить много элементов верстки.

И последнее, файл со скриптом respond.min.js подключается по условию(по условию могут также подключатсья и стили), чтобы этого добиться в WP есть 2 функции:

**wp\_script\_add\_data()** - Добавляет данные к подключенным на странице скриптам с помощью wp\_enqueue\_script(). Используйте когда, например нужно выводить скрипт только для «IE 6» или «lt IE 9» (меньше IE 9)... - <https://wp-kama.ru/function/wp_script_add_data>

**wp\_script\_add\_style()** - аналогичная функция - <https://wp-kama.ru/function/wp_style_add_data>

сразу после подключения

wp\_enqueue\_script( 'law-respond-js', get\_template\_directory\_uri() . '/assets/js/respond.min.js' );

вызываем:

wp\_script\_add\_data('law-respond-js', 'conditional', 'lt IE 9');

если посмотреть код страницы мы увидим:

<!--[if lt IE 9]>

<script type='text/javascript' src='http://law.loc/wp-content/themes/law/assets/js/respond.min.js?ver=5.2.2'></script>

<![endif]-->

**Урок 3 - Плагин TGM**

В данном уроке подключим Unyson.   
Ранее во второй части данного курса автор упоминал, что при установки темы пользователем, теме могут быть необходимы различные плагины для корректной работы ( + это требование Forest Team для premium тем ) и чтобы не писать мануал для польз. какие плагины необходимы для корректн. работы, как эти плагины установить, так как это далеко не самая лучшая практика, нам будет необходим плагин:

TGM Plugin Activation: <http://tgmpluginactivation.com/download/>

Инструкция по работе с TGM: <http://oddstyle.ru/wordpress-2/wordpress-plaginy/komanda-stoyashhaya-za-tgm-plugin-activation-vypustila-custom-generator.html>

Данный плагин проверяет установлен ли плагин из списка, который мы указали в настройках. Если какой-либо плагин, из списка, не найдено, то TGM выдаст предупреждение, либо требование об установке данного плагина, в зависимости от того, что указано в настройках.

Скачиваем TGM -> 

скачиваем и распаковываем плагин из архива в папку   
/law/inc  
и переименовываем /inc в /tgm

На что стоит обратить внимание, это:  
class-tgm-plugin-activation.php - класс который необходимо подключить.  
example.php - пример подключения

создаем файл tgm.php(Имя любое) и копируем в него содержимое из файла example.php из корневой папки плагина tgm

Подключаем tgm.php в functions.php в самом конце:

/\* TGM plugin \*/  
**require\_once** get\_template\_directory() . '/inc/tgm/tgm.php';

Затем в файле tgm.php указываем путь к классу class-tgm-plugin-activation.php:

**require\_once** get\_template\_directory() . '/inc/tgm/class-tgm-plugin-activation.php';

Далее настроим Array of plugin arrays в файле tgm.php,  
Array of plugin arrays - это массив который содержит массивы плагинов, в каждом подмассиве описывается отдельный плагин и способ его подключения.

tgm может устанавливать плагин как из оф.репозитрия ( это требование TF для premium тем ) так и из локального хранилища:  
law\inc\tgm\plugins  
в которое плагины помещаются в виде архива .zip

Удалим из law\_register\_required\_plugins() все не нужные плагины и оставим вот так:

/\*  
 \* Array of plugin arrays. Required keys are name and slug.  
 \* If the source is NOT from the .org repo, then source is also required.  
 \*/  
$plugins = **array**(  
 // This is an example of how to include a plugin from the WordPress Plugin Repository.  
 **array**(  
 'name' => 'Unyson', // Название плагина, можно посмотреть на https://ru.wordpress.org/plugins/  
 'slug' => 'unyson', // slug плагина (название папки в которой лежит сам плагин)  
 'required' => **false**, // false - предупреждение, true - требование об установке плагина  
 ),  
);

Согласно требованиям TM: 'required' => true, не допускается, разрешено только 'required' => false

Так же, ниже, в разделе

/\*  
 \* Array of configuration settings. Amend each line as needed.

В массиве $config = **array**( . . . .

Возможно нас заинтересует опция:

'is\_automatic' => **false**, // Automatically activate plugins after installation or not.

*Согласно требованиям TM: 'is\_automatic' => true, не допускается, но это надо проверить.*

Теперь если обновить страницу админки мы увидим предупреждение о том, что необходимо установить плагин Unyson:



так как мы его еще не установили.

Чтобы перейти к списку плагинов которые необходимо установить кликаем по 'Begin installing plugin' затем Install

Далее необходимо активировать плагин кликнув по 'Return to Required Plugins Installer' -> active плагин

Затем ставим расширения:  
Unyson -> Конструктор страниц - загрузить  
Unyson -> Слайдер - загрузить

По итогу: таким образом можно предлагать пользователю темы к установке, без которых тема будет работать не корректно.

**Урок 4 - Меню в шапке**

В текущем уроке реализуем шапку, которая включает в себя логотип и меню. Две отдельные кнопки Login и Sign Up представленны отдельной страницой в WP.

Начнем с определениея главной страницы:

в админке, страницы -> добавить новую: Home, в качестве теста укажем в контенте страницы 'Главная страницы' -> Опубликовать.

Создадим еще одну страницу для вывода последних записей, назовем ее Blog -> Опубликовать (пустую страницу).

Настройки -> чтение -> выбираем статическая стр. - Главная страница: Home, Страница записей: Blog -> сохранить изменения.

Теперь создадим шаблон front-page.php для главной страницы.

Далее необходимо начать резделять шаблон на логические части, начнем с header:

удаляем из header.php ненужные части и вставляем кусок шаблона из index.php в header.php:

<div id="page">  
 <nav class="fh5co-nav" role="navigation" ...>

Затем из index.php забираем начиная с элемента <footer ...> и до конца и переносим в footer.php, предварительно удалив из footer.php содержимое.

header и footer из index.php необходимо удалить соответственно.

Далее во front-page.php подключаем header и footer, а между ними обычный цикл WP который будет что-то выводить:

**<?php** get\_header(); **?>  
  
  
<?php if** ( have\_posts() ) : **while**( have\_posts() ) : the\_post(); **?>  
 <?php** the\_content(); **?>  
<?php endwhile**; **?>** <!-- post navigation -->  
**<?php else**: **?>** <!-- no posts found -->  
**<?php endif**; **?>  
  
  
<?php** get\_footer(); **?>**

Возвращаемся к header:

Что касается меню, то их необходимо 2 шт, одно для хэдер другое для футер. По умолчанию в файле functions.php в функции register\_nav\_menus() подключено только одно меню, нам необходимо зарегистрировать еще одно меню и переименовать меню по умолч:

// This theme uses wp\_nav\_menu() in one location.  
register\_nav\_menus( **array**(  
 'header-menu' => esc\_html\_\_( 'Header Menu', 'law' ),  
 'footer-menu' => esc\_html\_\_( 'Footer Menu', 'law' )  
) );

Затем заменяем исходное меню (теги <ul>и все что внутри них</ul>) из шаблона header.php вызовом функции wp\_nav\_menu():

**<?php** wp\_nav\_menu( **array**(  
 'theme\_location' => 'header-menu' // Указываем какое меню хотим вывести  
) ); **?>**

Затем в админке создаем само меню (наполняем его пунктами):

* Home - будет произвольной ссылкой
* Practice Areas - страница
* Won Cases - страница
* Blog - Это тоже страница, но в ней 4 рубрики:

Web Design  
eCommerce  
Branding  
API

* About - страница
* Contact - страница

Сначала в админке создадим 4 рубрики: Web Design, eCommerce, Branding, API (ярлык указывать не надо, он проставится автоматически)

В качестве тестовых данных создадим запись, которую поместим во все рубрики, кроме 'без рубрики' -> Опубликовать.

Теперь создадим странички для: Practice Areas, Won Cases, About, Contact.

Переходим в раздел внешний вид -> меню - и добавим новое меню, с названием 'Header Menu' -> создать меню -> добавляем пункты в Header Menu:



-> сохранить меню.

Далее необходимо добавить еще один аргумент в wp\_nav\_menu(), чтобы отменить контейнер оболочку для меню, по умолчанию, в качестве такого контнейнера выступает <div>, но у нас уже имеется свой контейнер:

<div class="col-xs-10 text-right menu-1">

**<?php** wp\_nav\_menu( **array**(  
 'theme\_location' => 'header-menu', // Указываем какое меню хотим вывести  
 'container' => **false** // Отменяем контейнер оболочку для меню  
 ) ); **?>**

</div>

Теперь необходимо разобраться со вложенными пунктами меню в пункте Blog:

WP генерирует свои классы для меню, для вложенных пунктов меню это будет класс 'sub-menu', а нам, согласно исходному макету, необходимо указать класс 'dropdown' для вложенных пунктов.  
И второе - для родительского класса, этих под пунктов меню, WP генерит класс menu-item-has-children, а нам, согласно исходному макету, необходимо указать класс 'has-dropdown'

Есть 2 варианта как этого добиться:

Первый - это найти в стилях темы классы 'dropdown' и 'has-dropdown' и переопределить их названия на соответсвующие названия из WP ('menu-item-has-children' и 'sub-menu')

Второй - это использ. аргумента $walker в wp\_nav\_menus(),

<https://youtu.be/Njx6oFz6B0c?list=PLV2Ofep-dtIzvQiV8u-MXV4JD57XyvlmT&t=1099>

в этом аргументе мы передаем объект класса:   
**Law\_Header\_Menu extends Walker\_Nav\_Menu**  
в котором мы переопределяем элемент меню и уровень его вложенности:

**<?php** wp\_nav\_menu( **array**(  
 'theme\_location' => 'header-menu', // Указываем какое меню хотим вывести  
 'container' => **false**, // Отменяем контейнер оболочку для меню  
 'walker' => **new** Law\_Header\_Menu // переопределяем элемент меню и уровень его вложенности  
) ); **?>**

Далее необходимо определить сам класс Law\_Header\_Menu:

**class** Law\_Header\_Menu **extends** Walker\_Nav\_Menu  
{  
 //code...  
}

затем класс Law\_Header\_Menu необходимо подключить в начале functions.php:

/\* вместо get\_template\_directory() можно использовать константу \_\_DIR\_\_\*/  
**require\_once** get\_template\_directory() . '/Law\_Header\_Menu.php';

После открываем код класса Walker\_Nav\_Menu, и копирум метод **start\_el()** в класс Law\_Header\_Menu и переопределяем необходимые нам параметры, такие как например $classes[] - массив классов;   
$classes[] формируется при помощи элементов массива $item, чтобы просмотреть содержимое этого массива есть одно удобное средство:

В ф-ке Unyson есть встроенная функция debug() (ранее мы писали подобную ф-ю сами) она удобна для вывода элементов из $item.   
Данная ф-я доступна из любого места в WP проекте, при условии, что подключен ф-к Unyson.

debug($item);

Если элемент меню явл. активным, это значит, что его [current] => true( это можно посмотреть вызвав debug($item) ), основываясь на данной инфе оформим активный пункт меню Home в Header Menu, присвоим ему класс 'active' (согласно исходному шаблону этот класс отвечает за актив. пункт меню), для этого в метод **start\_el()** добавляем следующее:

**if** ( $item->current ) {  
 $classes[] = 'active';  
}

Затем пункту меню <li>Blog <ul>...</ul></li>, который содержит подменю, необходимо присвоить класс 'has-dropdown', для этого надо воспольз. параметром $args функции start\_el():

**if** ( $args->walker->has\_children ) {  
 $classes[] = 'has-dropdown';  
}

И на последок необходимо заменить класс 'sub-menu' на 'dropdown', для этого мы обращаемся к методу

start\_lvl() - публикует узлы меню(теги <ul>),  
также копируем его в класс Law\_Header\_Menu и переопределяем элемент массива $classes[] на другой класс:

// Default class.  
$classes = **array**( 'dropdown' );

После всего выше проделанного, менюшка приобретет стилизваний вид, а так же заработает подсветка для активного пункта меню.

Но если зайти в пункт меню Blog, то мы увидим одну недоработку с версткой активного пункта меню, чтобы это исправить поправим файл wp- /assets/css/style.css

/\* my new menu rules\*/  
.fh5co-nav ul.dropdown > li.active > a {  
 color: #f95959 **!important**;  
 position: relative;  
}  
  
.fh5co-nav ul.menu > li.active > a:after {  
 position: absolute;  
 bottom: 25px;  
 left: 15px;  
 content: '';  
 width: 20px;  
 height: 2px;  
 background: #F95959;  
 margin: 0 auto;  
}  
  
/\*.fh5co-nav ul li.active > a:after {\*/  
 /\*position: absolute;\*/  
 /\*bottom: 25px;\*/  
 /\*left: 15px;\*/  
 /\*content: '';\*/  
 /\*width: 20px;\*/  
 /\*height: 2px;\*/  
 /\*background: #F95959;\*/  
 /\*margin: 0 auto;\*/  
/\*}\*/

После добавления данных стилей, если теперь пройти по какому-либо из пунктов подменю, то при последующем наведение на Blog, активный пункт подменю будет выделен красный шрифтом.

Единственный косяк который осталось исправить, это - если пройти по пункту меню Blog, то подгрузится индексная страничка(шаблон index.php), поэтому в index.php убираем header часть(*которую мы забыли убрать ранее. Странно почему автор стразу не вырезал эти части шаблона во время перенса.*) и пишем вместо нее get\_header();

Теперь пункт Blog если он активный подсвеч. корректно.

??? Но есть одно место которое мне не нравится, если перейти по пункту подменю, то подсветка активного пункта подменю будет видна только если навести на Blog, в остальное время, когда подменю не видно Blog не подсвечивается и становится не понятно где мы находимся если ориентироваться только на меню, надо это как-то решит.

**Урок 5 - Логотип сайта**

Выводим лого динамически, и займемся красной точкой.

Находим <div> с лого в header.php, указываем

<div id="fh5co-logo"><a href="**<?php echo** home\_url('/'); **?>**">**<?php** bloginfo('name'); **?>**</a></div>

В кастомайзере дописываем после Law '.' точку.

Затем эту точку надо заключ. в <span></span> и окрасить в красный цвет, средствами JS.   
Допишем немного js кода в начале скрипта main.js:

**var** str = $('#fh5co-logo a').text();  
str = str.slice(0, -1) + '<span>' + str.substr(-1) + '</span>';  
 $('#fh5co-logo a').html(str);

Теперь точка после Law, в лого, отображается красным цветом, согласно макету.

**Урок 6 - Кастомизация слайдера фреймворка**

В Unyson имеется 3 вида слайдеров: bx-slider, niva-slider, owl-carousel.

Данные слайдеры можно кастомизировать, напрмер через стили и js

Но в данном уроке мы будем рассматривать как добавить новый вариант слайдера, сделанный вручную.

оф.док. раздел по сладйеру: <http://manual.unyson.io/en/latest/extension/slider/index.html>

Для начала создадим директорию для кастомизированного слайдера(*см. также Часть 2 Урок 8 - Секция Team. часть 1*)  
В папку с самой темой необходимо поместить след. структуру из директорий:

framework-customizations/extensions/framework/extensions/media/extensions/slider/extensions

Так как у нас уже имеются кастомизированные элементы Unyson и соответсвенно у нас уже создана структура из директорий, хотя и не совсем походящая, в этом случае мы берем часть необходимой структуры и добавляем в уже имеющуюся, т.е. в папку /extensions кладем:

/law/framework-customizations/extensions/media/extensions/slider/extensions/<папка со слайдером для кастомизации>

Затем копирируем туда из самого ф-ка bx-slider(так как он наиболее расширенный) и переименовывем папку с ним во /flex-slider,  
также необходимо перименовать и сам файл в папке /views, во   
flex-slider.php

Из папки /flex-slider нам потребуются следующие файлы: manifest.php в нем оставляем только следующее:

**<?php if** (!defined('FW')) **die**('Forbidden'); // Чтобы не было прямого обращения к файлу  
  
$manifest['standalone'] = **true**;

Этого будет вполне достаточно.

В config.php необходимо следующее

**<?php if** (!defined('FW')) **die**('Forbidden');  
/\* откуда будем брать картинки  
 \* Из 'posts', 'categories', 'tags' или 'custom'  
 \* 'custom' - означает, что будем брать вручную \*/  
$cfg['population\_methods'] = **array**('custom');  
  
/\* Типы контента отображаемый слайдером \*/  
$cfg['multimedia\_types'] = **array**('image');

Пока что views/flex-slider.php оставляем только следующее:

**<?php if** (!defined('FW')) **die**('Forbidden'); **?>  
<?php** debug($data['slides']); **?>**

debug($data['slides']) - для разработчика

В папке /options лежат файлы в которых мы настраиваем как мы будем выбирать картинки, в данном проекте требуется оставить только custom.php, остальное удаляем.

Далее нас интересует /static, в ней лежат скрипты и стили.  
Удаляем содержимое папок /css и /js, так как в данной теме это не требуется.

preview.jpg и thumb.jpg в папке /images это картинки которые будут показаны в preview слайдера в админке.

Заменяем две исходные картинки preview.jpg и thumb.jpg в папке   
law\framework-customizations\extensions\media\extensions\slider\extensions\flex-slider\static\images   
картинками из урока.

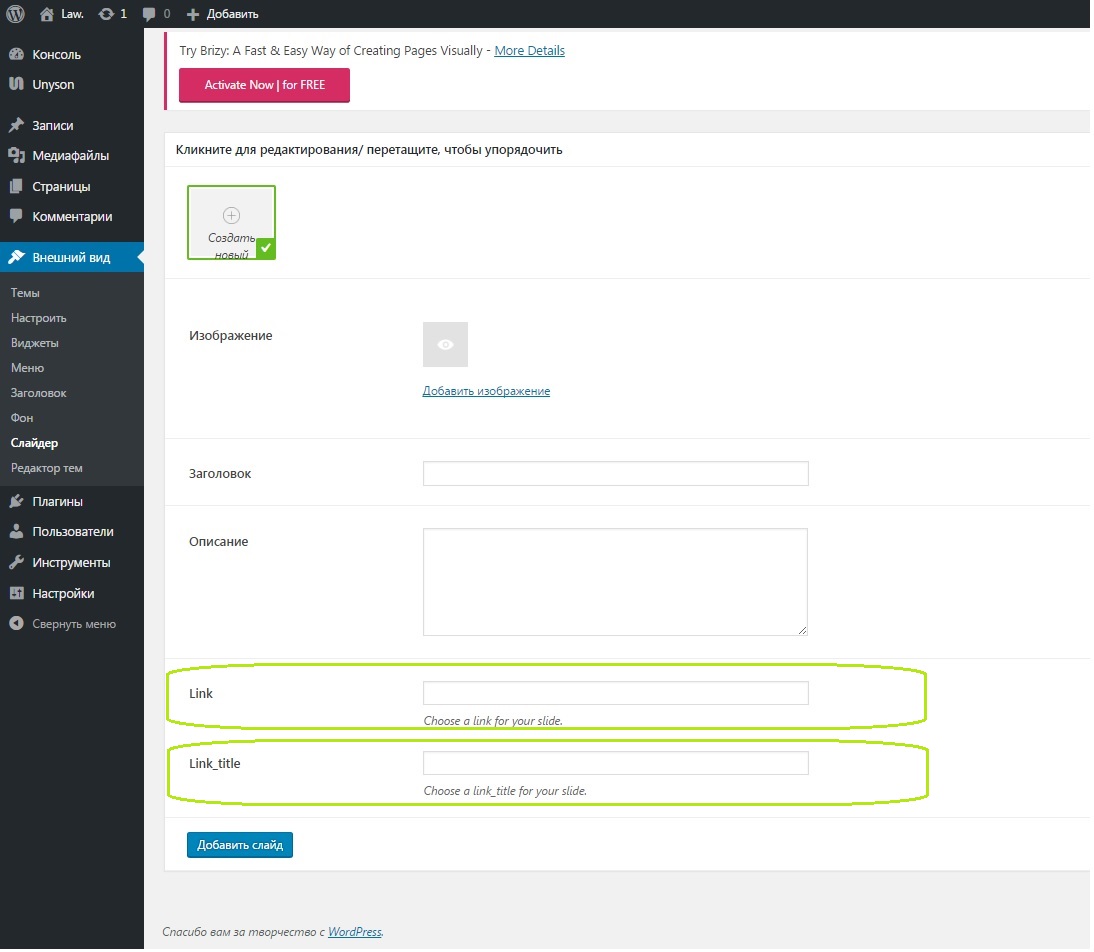
Теперь если в админке зайти во внеш.вид -> слайдеры, то мы увидим помимо трех слайдеров по умолч., еще и наш 4й слайдер.

Создадим заголовок для слайдера:  
внеш.вид -> слайдер -> в поле заголовок пишем ('Home') -> создать.

Затем кликаем на 'Добавить заголовок' -> Здесь укажем заголовок (Expert Legal Solutions) и описание(Free html5 templates Made by freehtml5.co),  
но перед тем как выводит кнопку для нее небходимо создать 2 опции в custom.php:

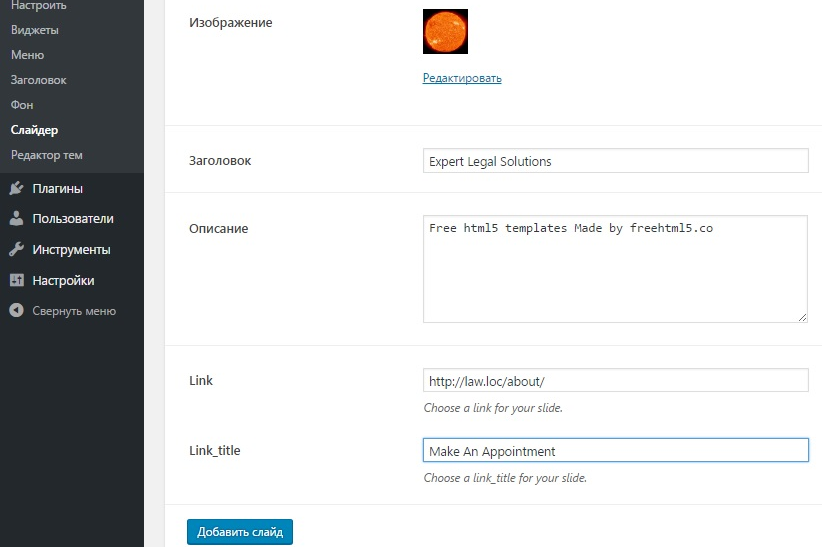
**<?php if** ( ! defined( 'FW' ) ) {  
 **die**( 'Forbidden' );  
}  
$options = **array**(  
 'link' => **array**( // сама ссылка  
 'type' => 'text',  
 'label' => \_\_( 'Link', 'law' ),  
 'value' => '',  
 'desc' => \_\_( 'Choose a link for your slide.', 'law' )  
 ),  
 /\* добавим еще одну настройку \*/  
 'link\_title' => **array**( // текст ссылки  
 'type' => 'text',  
 'label' => \_\_( 'Link\_title', 'law' ),  
 'value' => '',  
 'desc' => \_\_( 'Choose a link\_title for your slide.', 'law' )  
 )  
);

Вот так создаются доп.опции, после этого появится 2 новых настройки:



Затем добавляем картинки для слайдера в медиафайлы -> добавить новый.

внеш.вид -> слайдер -> кликаем на Home Slider -> создать новый -> далее заполняем поля:



-> Добавить слайд.

по аналогии добавим еще пару картинок в слайдер(ссылки добавлять не будем) -> сохранить.

Слайдер готов, теперь его можно вставить на страничку под именем Home Slider.

Переходим к стр. Home -> unyson builder -> создаем секцию(в настройках на всю ширину - да, id = home-slider) и из мультимедиа добавляем слайдер указав в настройках имя слайдера - Home Slider(ширину высоту убираем) -> сохранить -> обновить.

Теперь обновив стр. мы увидим информацию о картинках в слайдере, благодаря ранее вызванной ф-ции debug() в файле flex-slider.php

Чтобы слайдер окончательно заработал, перепишем файл flex-slider.php  
Вместо debug пишем if и в него вставляем <aside> со слайдером из index.html. Затем оставляем только один <li> и помешаем его в цикл foreach(). Также необходимо заменить некоторые html атрибуты:

**<?php if** (!defined('FW')) **die**('Forbidden'); **?>  
  
<?php if**( $data['slides'] ): **?>** <aside id="fh5co-hero" class="js-fullheight">  
 <div class="flexslider js-fullheight">  
 <ul class="slides">  
 **<?php foreach** ($data['slides'] **as** $slide): **?>** <li style="background-image: url(**<?php echo** $slide['src']; **?>**);">  
 <div class="overlay-gradient"></div>  
 <div class="container">  
 <div class="row">  
 <div class="col-md-8 col-md-offset-2 text-center js-fullheight slider-text">  
 <div class="slider-text-inner">  
 <h1>**<?php echo** $slide['title']; **?>**</h1>  
 <h2>**<?php echo** $slide['desc']; **?>**</h2>  
 <!-- ссылка может быть прописана, а может и нет, поэтому использ. условие -->  
 **<?php if**( !**empty**( $slide['extra']['link'] ) && !**empty**( $slide['extra']['link\_title'] ) ): **?>** <p><a class="btn btn-primary btn-lg" href="**<?php echo** $slide['extra']['link']; **?>**">**<?php echo** $slide['extra']['link\_title']; **?>**</a></p>  
 **<?php endif**; **?>** </div>  
 </div>  
 </div>  
 </div>  
 </li>  
 **<?php endforeach**; **?>** </ul>  
 </div>  
 </aside>  
**<?php endif**; **?>**

Теперь на главной появится слайдер.

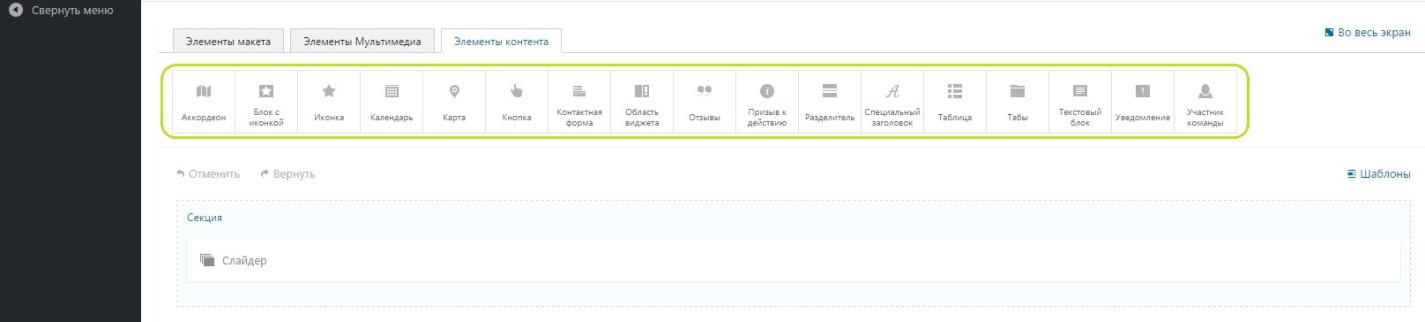
Немного подправим его стили чтобы он соотвествовал исходному макету.

Все управление слайдером производится через внеш.вид -> слайдер -> заходим в нужный слайдер ...

**Урок 7 - Пользовательский шорткод для Unyson. Часть 1**

В этом уроке будем делать секцию со счетчиками. Встроенными средствами Unyson, реализовать такое трудно, поэтому кастомизируем его под свои нужды.   
Для этого создадим свой компонент(кастомный shortcode) - который назовем span(т.е. это будет тег <span>). Этот компонент span будет принимать следующие аргументы: значения для различных атрибутов(например: id, class и т.д.), текстовое содержимое самого span и иконку которая будет помещена внутри span, так как встроенный shortcode иконки не подойдет.

*рис. shortcodes идущие по умолч. в Unyson page builder*



За основу при создании кастомного shortcode для span возьмем компонент icon(который мы ранее кастомизировали):  
**/law/framework-customizations/extensions/shortcodes/shortcodes/icon**  
копируем его и переименовываем в /span.  
Вообще все shortcodes имеют схожую структуру(теже папки и файлы).

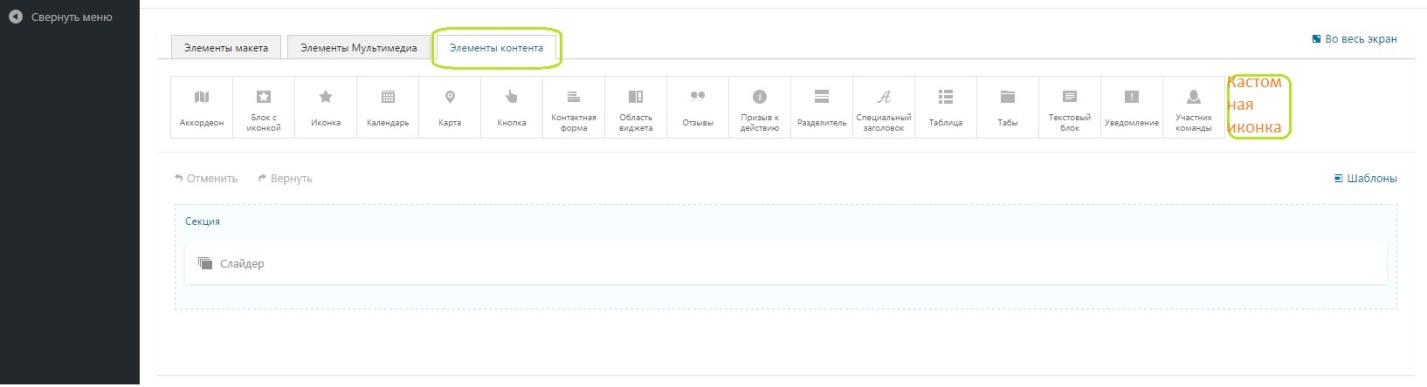
Начнем кастомизировать элемент span, немного подправим файл static.php, так как стили вывода элемента в админку не имеет смысл менять, это значит, что мы оставляем подключение стилей в static.php и сами стили(из папки css) тоже.   
Единственное, что необходимо поменять это handle на 'fw-shortcode-span' и поменять путь откуда брать css файл:

$shortcodes\_extension = fw\_ext('shortcodes');  
wp\_enqueue\_style(  
 'fw-shortcode-span',  
 $shortcodes\_extension->get\_declared\_URI('/shortcodes/span/static/css/styles.css'),  
 **array**('font-awesome')  
);

**Идем дальше, следующий файл это** **config.php**, в нем определяется как будет подписана иконка, что будет отображаться в всплывающей подсказке

$cfg['page\_builder'] = **array**(  
 'title' => \_\_('Span', 'law'),  
 'description' => \_\_('Add a Span', 'law'),  
  
 /\*  
 \* Указываем где будет расположена иконка(в какой вкладке)  
 \* без domain 'fw' работает криво, создает новый tab  
 \* даже если попытаться поместить икноку в уже сущ-ий tab  
 \* \*/  
 'tab' => \_\_('Content Elements', 'fw'),  
);

В параметре 'tab' указываем в какой вкладке мы хотим отобразить нашу кастомную иконку, обычно помещают в Элементы мультимедиа:



Но мы также можем создавать и свои вкладки(см.оф.док).

**В** **options.php** **описаны** сущ-ие поля(настройки), там же описываем новые поля:

**<?php if** ( ! defined( 'FW' ) ) {  
 **die**( 'Forbidden' );  
}  
  
$options = **array**(  
 'span\_icon' => **array**( // Поле для добавления иконки  
 'type' => 'icon',  
 'label' => \_\_( 'Span Icon', 'law' )  
 ),  
 'span\_content' => **array**( // содержимое внутри тега <span>...</span>  
 'type' => 'text',  
 'label' => \_\_( 'Span Content', 'law' ),  
 'desc' => \_\_( 'Span Content Desc', 'law' ),  
 ),  
  
 'span\_class' => **array**( // класс тега <span>  
 'type' => 'text',  
 'label' => \_\_('Span Class', 'law'),  
 'desc' => \_\_( 'Span Class Desc', 'law' )  
 ),  
  
 'span\_id' => **array**( // ID тега <span>  
 'type' => 'text',  
 'label' => \_\_('Span ID', 'law'),  
 'desc' => \_\_( 'Span ID Desc', 'law' )  
 ),  
  
 /\* здесь описываем data атрибуты для тега span  
 \* Поле принимает строку в виде:  
 \* data-from="0" data-to="27539" data-speed="5000" data-refresh interval="50"  
 \*/  
 'span\_data\_attr' => **array**(  
 'type' => 'text',  
 'label' => \_\_('Attributes', 'law'),  
 'desc' => \_\_( 'Attributes Desc', 'law' )  
 )

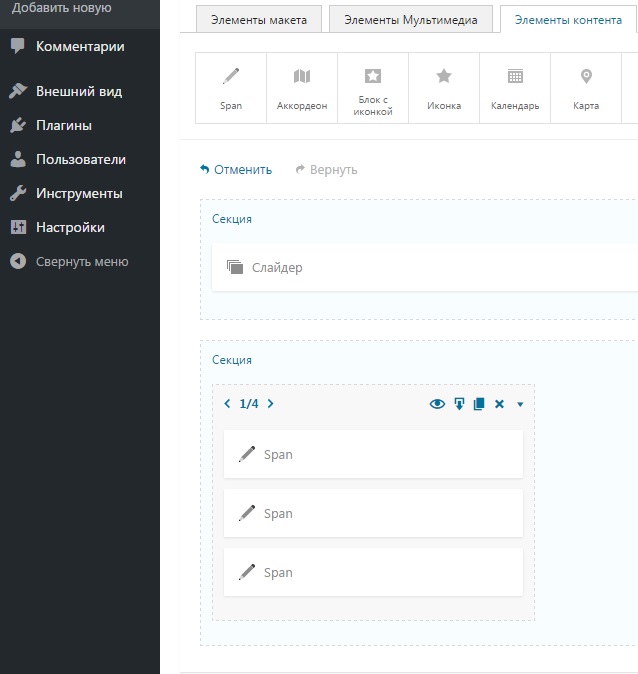
);

**Идем дальше, папка static/img/** содержит картинку, которая будет показана в админке, в page builder, картинка обязательно должна называться page\_builder.png и находится в директории static/img/

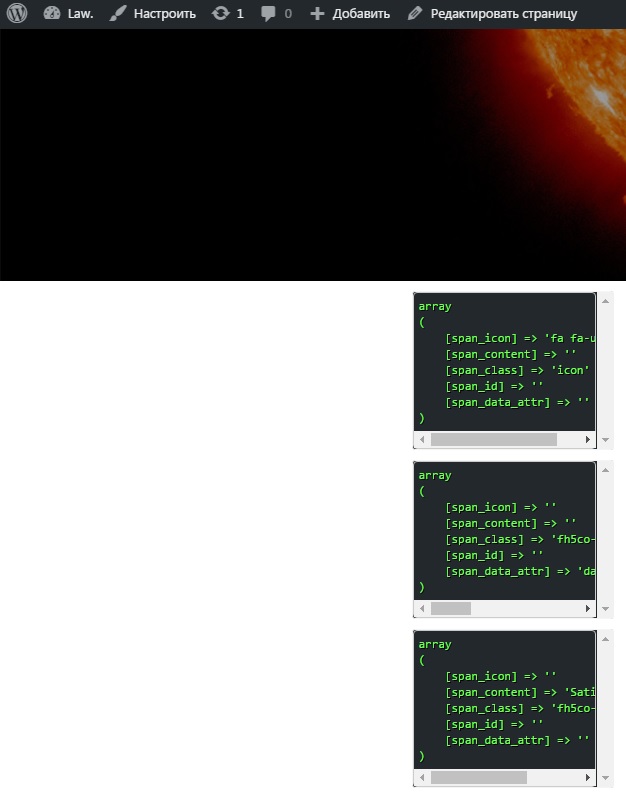
**Далее идет файл view.php**

В данном файле мы работаем с массивом $atts[], что бы просмотреть его содержимое использ. debug();  
Пока во view.php вместо шаблона вывода иконки оставим просто debug();

Затем продолжаем конструировать страницу home в page builder,  
1й - span содержит иконку, 2й - скрипт счетчика, 3й - содержимое тега span:



Теперь на стр. Home, под слайдером мы увидим результат работы debug() из view.php.   
В каждом массиве содержутся, созданные нами ранее в options.php, значения аттрибутов для каждого из 3х тегов span, которые мы указали в админке.



Продолжение в след. уроке ...