### 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.

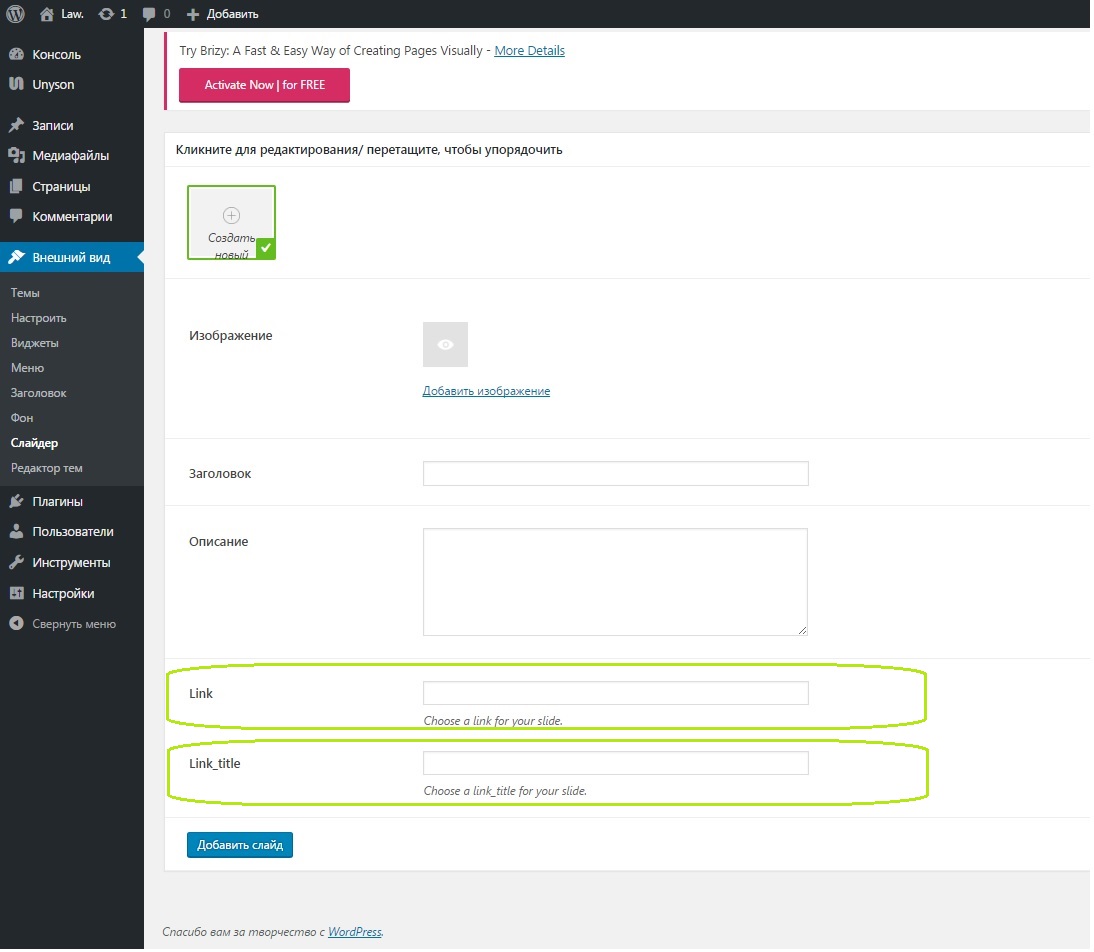
Теперь если в админке зайти во внеш.вид -> слайдеры, то мы увидим помимо трех слайдеров по умолч., еще и наш 4й слайдер.

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:

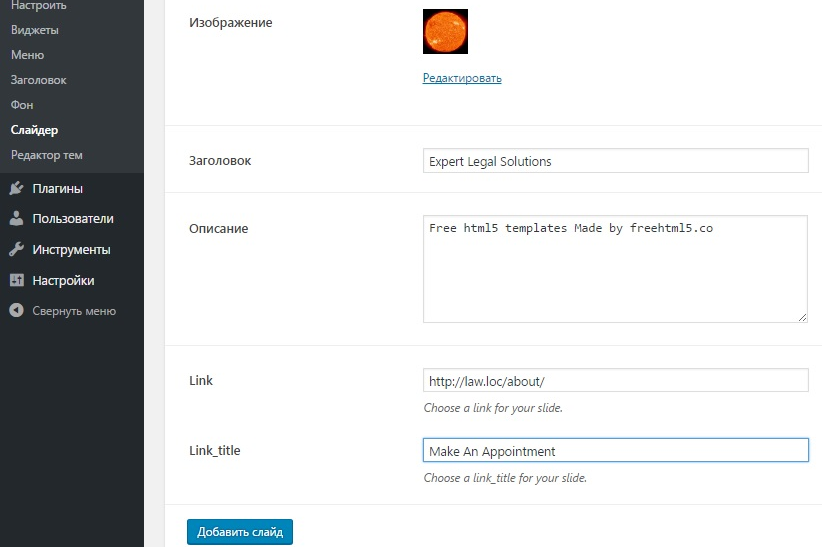
**<?php if** ( ! defined( 'FW' ) ) {  
 **die**( 'Forbidden' );  
}  
$options = **array**(  
 'link' => **array**( // сама ссылка  
 'type' => 'text',  
 'label' => \_\_( 'Link', 'law' ),  
 'value' => '',  
 'desc' => \_\_( 'Choose a link for your slide.', 'law' )  
 ),  
 /\* добавим еще одну настройку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

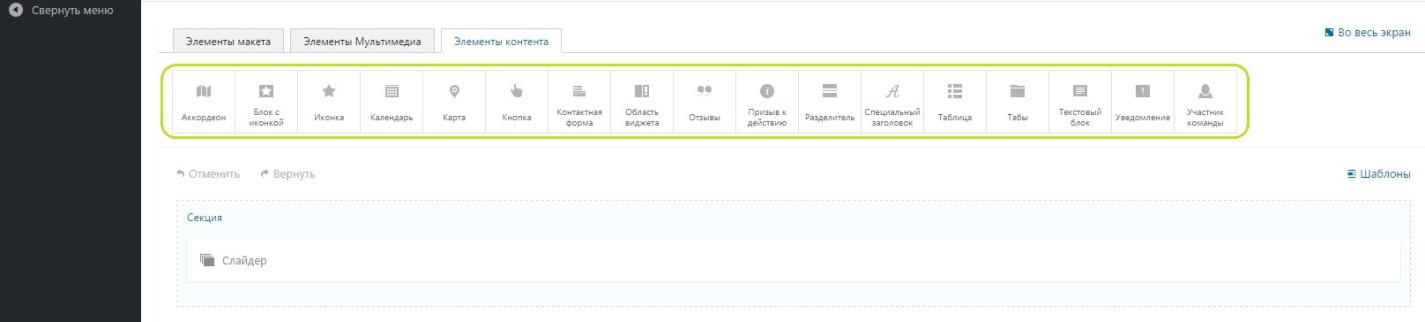
Немного подправим его стили чтобы он соотвествовал исходному макету.

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).

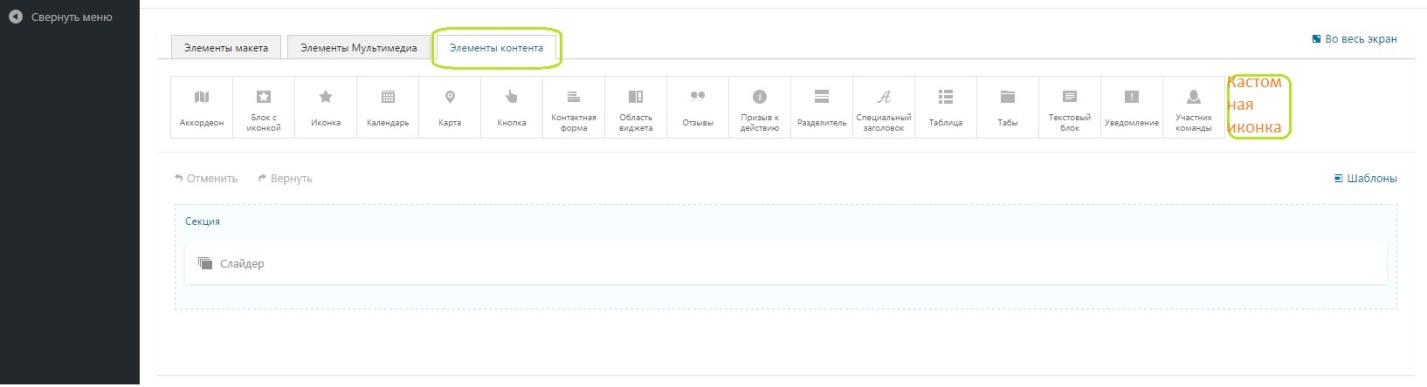
Начнем кастомизировать элемент span, немного подправим файл static.php, так как стили вывода элемента в админку не имеет смысл менять, это значит, что мы оставляем подключение стилей в static.php и сами стили(из папки css) тоже.   
Единственное, что необходимо поменять это handle на 'fw-shortcode-span'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

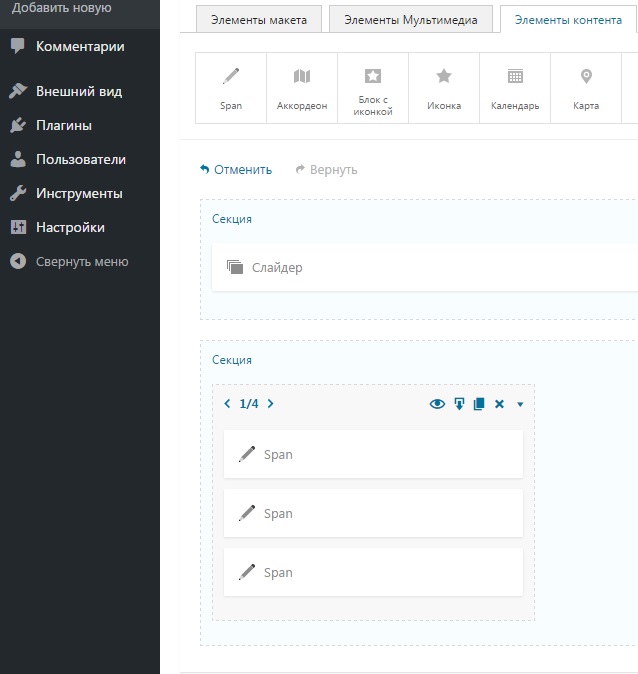
);

**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/

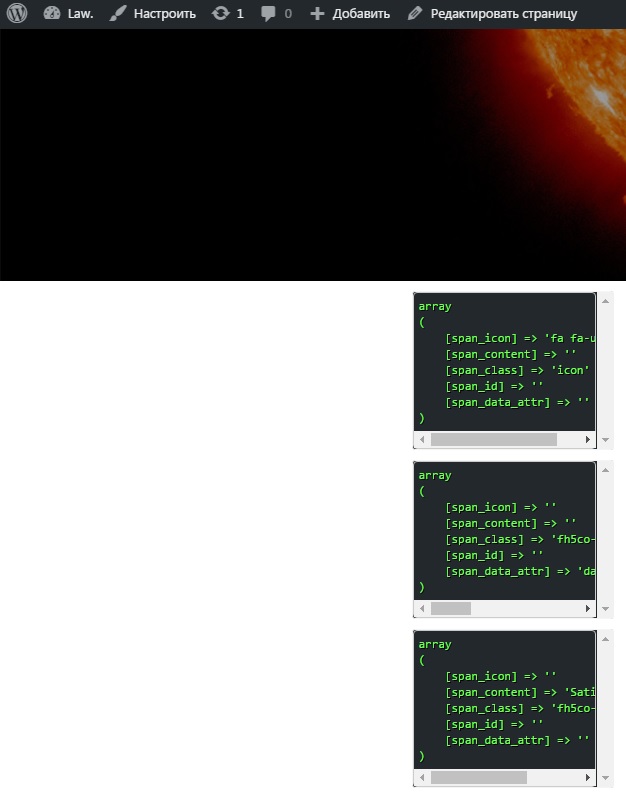
**Далее идет файл view.php**

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 ...

**Урок 8 - Пользовательский шорткод для Unyson. Часть 2**

Опишем view.php для компонента span:

<span  
 **<?php if** ( !**empty**( $atts['span\_class'] ) ) { **echo** "class='{$atts['span\_class']}'"; } **?>  
 <?php if** ( !**empty**( $atts['span\_id'] ) ) { **echo** "id='{$atts['span\_id']}'"; } **?>  
 <?php if** ( !**empty**( $atts['span\_data\_attrs'] ) ) { **echo** $atts['span\_data\_attrs']; } **?>**>  
 **<?php if** ( !**empty**( $atts['span\_icon'] ) ) : **?>** <i class="**<?php echo** $atts['span\_icon']; **?>**"></i>  
 **<?php endif**; **?>  
  
 <?php if** ( !**empty**( $atts['span\_content'] ) ) **echo** $atts['span\_content']; **?>**</span>

Теперь необходимо настроить стили, для этого добавим необходимые классы.

Для секции добавляем класс 'fh5co-counters' fh5co-bg-section, фон.цвет #f2f2f2, на всю шир. - нет., ID - 'fh5co-counter'

Еще на случай когда ID для секции не указан, добавим условие по которому в этом случае пустой ID не будет указываться для секции.   
Для этого добавим условие в тег section в файле section/views/view.php:

<section **<?php if** ( ( **isset**( $atts['custom\_id'] ) ) && ( $atts['custom\_id'] ) ) **echo** "id='{$atts['custom\_id']}'"; **?>** class="fw-main-row **<?php echo** esc\_attr($section\_extra\_classes) **?>**" **<?php echo** $section\_style; **?> <?php echo** $bg\_video\_data\_attr; **?>**>  
 <div class="**<?php echo** esc\_attr($container\_class); **?>**">  
 **<?php echo** do\_shortcode( $content ); **?>** </div>  
</section>

-> сохранить -> обновить.

Далее для span которые содержат JS counter добавляем аттрибуты:  
data-from="0" data-to="27539" data-speed="5000" data-refresh interval="50"

Затем на <div> (колонка), который оборачивает наши 3 span(a) необходимо навесить класс 'animate-box', для того чтобы заработала анимация.   
А также необходимо добавить класс 'text-center', чтобы внутри колонки все было выровнено по центру.   
Сделать это можно 2мя путями:

1й вариант - задать класс 'text-center' для всей секции, т.е. передать в поле Custom Class для секции сразу несколько классов одной строкой:  
'fh5co-counters fh5co-bg-section text-center',  
но это не всегда удобно, плюс нам надо добавить анимацию именно для <div>, которые оборачивают span теги, а не для всей секции.

2й вариант() это использ. скрипта main.js, при помощи js найдем необходимые теги div и добавим необходимые классы:

//My js for counter columns  
$('#fh5co-counter > .fw-container > .fw-row > div').addClass('animate-box');  
// end

Анимация заработала, но мы пойдем другим путем.

3й вариант - Кастомизируем колонки. Добавим для колонок возможность добавлять свои классы и ID. Также как и в прошлый раз, берем за основу готовый компонент /column в корне ф-ка Unyson и копируем его в /framework-customizations/extensions/shortcodes/shortcodes/

В /column добавляем файл options.php, так как изначально его там нет, за основу можно взять любой другой уже сущ-ий options.php из другого компонента:

**<?php if** ( ! defined( 'FW' ) ) {  
 **die**( 'Forbidden' );  
}  
  
$options = **array**(  
 'col\_class' => **array**(  
 'type' => 'text',  
 'label' => \_\_( 'Column Class', 'fw' ),  
 'desc' => \_\_( 'Column Class', 'fw' )  
 ),  
 'col\_id' => **array**(  
 'type' => 'text',  
 'label' => \_\_( 'Column ID', 'fw' ),  
 'desc' => \_\_( 'Column ID', 'fw' ),  
 ),  
  
);

Далее допишем view.php:

**<?php if** (!defined('FW')) **die**('Forbidden');  
  
$class = fw\_ext\_builder\_get\_item\_width('page-builder', $atts['width'] . '/frontend\_class');  
**?>**<div class="**<?php echo** esc\_attr($class); **?><?php if** ( !**empty**($atts['col\_class']) ) { **echo** $atts['col\_class']; } **?>**"**<?php if** ( !**empty**($atts['col\_id']) ) { **echo** "id='{$atts['col\_id']}'"; } **?>**>  
 **<?php echo** do\_shortcode($content); **?>**</div>

Теперь в админке для кололонок можно задавать классы и ID.

Зададим классы 'text-center animate-box'

Дублируем колонки, и добавляем для каждой колонки свою иконку, а в span со счетчиком соответсвующие ему атрибуты.

**Урок 9 - Секция видео. Часть 1**

Данная секция занимает всю ширину экрана и делится на 2 части.

Вместо того чтобы видео проигрывалось в отдельном окне поверх основной страницы, сделаем так, чтобы оно проигрывалось в той же области в которой находится изначально. Помимо всего прочего видео должно быть адаптивным, сделаем это при помощи классов Bootstrap.

Открываем страницу Home в page builder -> добавим новую секцию -> добав. колонку 1/2 -> добав. в колонку видео -> в настройках компонента видео указываем ссылку на видео ресурс(\* как это сделать см. ниже), ширину-высоту убираем, так как видео должно быть адаптивным.

видео ресурс(\* как это сделать) - youtube -> поделиться -> встроить -> из предложенного html кода необходимо взять только ссылку:

<iframe width="560" height="315" src="https://www.youtube.com/embed/HFT7ATLQQx8" frameborder="0" allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>

Добавим ID для секции fh5co-content и выберем на всю ширину -> сохранить.

Следующий момент который необходимо исправить это ШхВ видео по умолчанию. Для этого кастомизируем компонент видео, техника все таже,  
копируем компонент /media-video из корня ф-ка Unyson из  
wp-content/plugins/unyson/framework/extensions/shortcodes/shortcodes/media-video/

в wp-content/themes/law/framework-customizations/extensions/shortcodes/shortcodes

Затем правим view.php, за основу для <div> обертки возьмем кусок шаблона из bootstrap, (<https://getbootstrap.com/docs/4.3/utilities/embed/>) - из Embeds:

Через админку добавим контент во вторую колонку, заголовок и абзац:

</h2> - Why Choose Us?

</p> - Far far away, behind the word mountains, far from the countries Vokalia and Consonantia, there live the blind texts far from the countries Vokalia and Consonantia, there live the blind texts.

Далее необходимо убрать отступы по краям, для этого необходимо добавить еще один селектор в /law/assets/css/law.css:

#home-slider > .fw-container-fluid,  
#fh5co-content > .fw-container-fluid {  
 padding: 0;  
}

**Урок 10 - Секция видео. Часть 2**

В этом уроке стилизуем правую часть данной секции.

Первое что необходимо сделать это добавить классы 'choose animate-box' и класс 'fh5co-heading' для колонки. Для этого в админке в настройках данной колонки в поле Custom class добавим классы 'choose animate-box fh5co-heading' -> сохранить -> обновить.

Затем немного подправим отступы для этого в assets/css/law.css добавим правило для padding в правой колонке:

#fh5co-content .choose {  
 padding: 20px 40px 0;  
}

Теперь необходимо реализовать прогресс бары. Для данной цели хорошо подойдет ранее созданный нами компонент span, добавим его в правую колонку. А также нашему span (который выступает в качестве div обертки согласно исх. шаблону) необходимо добавить классы:  
'progress-bar progress-bar-striped active'

Затем внести контент в поле Span Content: Adoption Law 50%

И после добавить атрибуты role="progressbar" aria-valuenow="50" aria-valuemin="0" aria-valuemax="100" style="width:50% - style="width это ширина progress bar, на сколько процентов он заполнен.

Progressbar появился, но для него необходимо добавить класс обертку, для этого добавим еще две кастомные настройки для компонента span, открываем options.php данного компонента и добавляем новые option:

$options = **array**(  
 'span\_wrapper' => **array**(  
 'type' => 'checkbox',  
 'label' => \_\_( 'Wrap in div?', 'law' )  
 ),  
  
 'span\_wrapper\_class' => **array**(  
 'type' => 'text',  
 'label' => \_\_( 'Wrapper Class', 'law' )  
 ),

. . . . .

. . . . .

. . . . .

Затем новые options необходимо описать во **view.php**:

**<?php if** ( !**empty**($atts['span\_wrapper'] ) ): **?>** <div **<?php if** ( !**empty**($atts['span\_wrapper\_class'] ) ) { **echo** "class='{$atts['span\_wrapper\_class']}'"; } **?>**>  
**<?php endif**; **?>** <span  
 **<?php if** ( !**empty**( $atts['span\_class'] ) ) { **echo** "class='{$atts['span\_class']}'"; } **?>  
 <?php if** ( !**empty**( $atts['span\_id'] ) ) { **echo** "id='{$atts['span\_id']}'"; } **?>  
 <?php if** ( !**empty**( $atts['span\_data\_attrs'] ) ) { **echo** $atts['span\_data\_attrs']; } **?>** >  
 **<?php if** ( !**empty**( $atts['span\_icon'] ) ) : **?>** <i class="**<?php echo** $atts['span\_icon']; **?>**"></i>  
 **<?php endif**; **?>  
  
 <?php if** ( !**empty**( $atts['span\_content'] ) ) **echo** $atts['span\_content'] **?>** </span>  
**<?php if** ( !**empty**($atts['span\_wrapper'] ) ): **?>** </div>  
**<?php endif**; **?>**

Далее через админку добавляем класс для <div> обертки:  
Wrap in div? - true   
Wrapper Class - progress  
сохранить -> обновить.

Осталось добавить 3 недостающих прогресс бара, добавляем по аналогии с первым:

Family Law 80%, Real Estate Law 70%, Personal Injury Law 40%

Далее подправим стили для прогресс баров, добавим правила в law.css:

#fh5co-content .fh5co-heading {  
 margin-bottom: 0;  
}  
  
.fh5co-heading h2 {  
 margin-bottom: 15px;  
}  
  
.fh5co-heading span {  
 display: block;  
 margin-bottom: 5px;  
 text-transform: none;  
 font-size: 12px;  
 letter-spacing: 0;  
}  
  
.progress {  
 margin-bottom: 15px;  
}

Прогресс бары готовы.

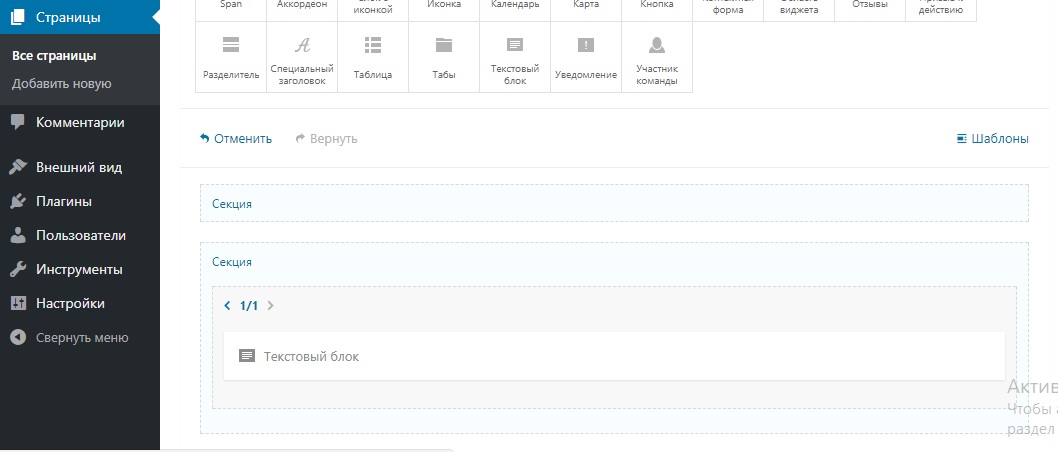
**Урок 11 - Секция услуг**

Создадим новую секцию в page builder.

Верхняя часть секции состоит из заголовка, который занимает 8 колонок, а слева и справа по 2 пустые (offset) колонки, к сожалению, Unyson не поддерживает работу с колонками offset.

Секция должна иметь класс 'fh5co-bg-section’ и id ‘fh5co-practice’, цвет bg #f2f2f2, на всю ширину – да.

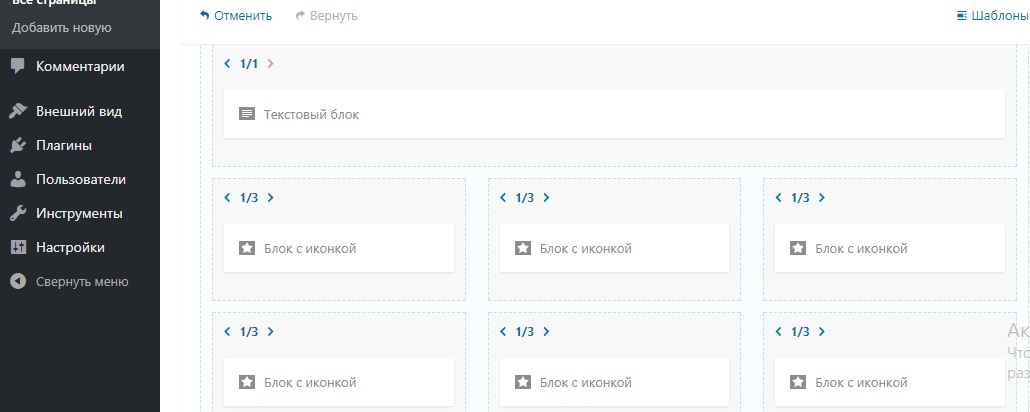
1я колонка(с заголовком) должна иметь классы ‘text-center fh5co-heading’.



Дальше необходимо отцентрировать текст под заголовком по центру.  
Для этого напишем правило CSS (немного копирующее сетку Bootstrap). Одна колонка занимает примерно 8,33…3% ширины row,  
Нам надо получить по 2 пустые колонки слева и справа, т.е. мы можем просто указать padding-left и padding-right по 16,66% (8,33 x 2) округляем до целого т.е. 16%.

Далее все также в Page builder реализуем колонки с иконками.

(Согласно исх. макету заголовок карточки должен быть ссылкой, для это необходимо самостоятельно кастомизировать колонку, добавив возможность оборачивать контент внутри h3 в <a>) – готово, см. icon-box

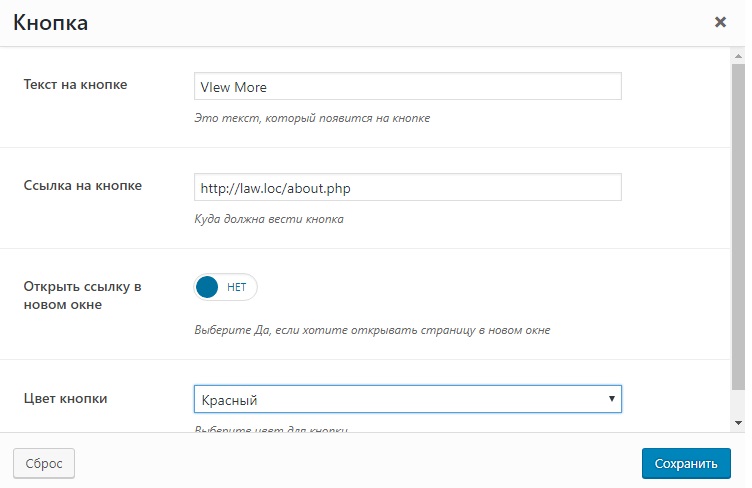
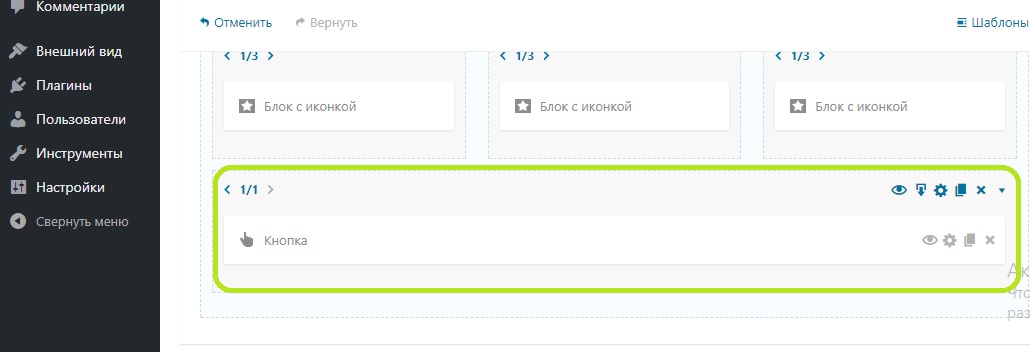


Затем разберемся со стилями, для этого при помощи собственного правила стилей law-services в файле law.css переопределим некоторые стили из класса ‘services’ для колонки.

Добавим класс ‘law-services’ и ‘animate-box’ и остальные правила стилей для колонки через админку.

.law-services .fw-iconbox {  
 position: relative;  
 padding: 30px;  
 margin-bottom: 60px;  
 background: #fff;  
 border: 1px solid rgba(0,0,0, 0.2);  
 transition: 0.3s;  
 border-radius: 4px;  
 box-shadow: 0px 2px 5px 0px rgba(0, 0, 0, 0.06);  
}  
.law-services .fw-iconbox h3 a {  
 color: #000;  
}  
.law-services .fw-iconbox .fw-iconbox-image {  
 width: 90px;  
 height: 90px;  
 background: #F95959;  
 display: table;  
 text-align: center;  
 margin: -75px auto 0 auto;  
 margin-bottom: 30px;  
 border-radius: 50%;  
 transition: 0.3s;  
}  
@media screen and (max-width: 992px) {  
 .law-services .fw-iconbox .fw-iconbox-image {  
 margin: 0 auto 30px auto;  
 }  
}  
.law-services .fw-iconbox .fw-iconbox-image i {  
 display: table-cell;  
 vertical-align: middle;  
 height: 90px;  
 font-size: 40px;  
 line-height: 40px;  
 color: #fff;  
}  
.law-services .fw-iconbox:hover, .law-services .fw-iconbox:focus {  
 box-shadow: 0px 14px 30px -15px rgba(0, 0, 0, 0.75);  
}  
.law-services .fw-iconbox:hover .fw-iconbox-image, .law-services .fw-iconbox:focus .fw-iconbox-image {  
 background: #007CB9;  
 box-shadow: 0px 5px 0px 0px rgba(0, 0, 0, 0.13);  
}  
.law-services .fw-iconbox .fw-iconbox-title h3 {  
 transition: .5s;  
 font-weight: 400;  
}  
.law-services .fw-iconbox:hover h3, .services:focus h3 {  
 color: #F95959;  
}

И последнее что осталось сделать это кнопка:



Стили для кнопки:

#fh5co-practice .fw-btn{  
 background-color: #F95959;  
 border: 2px solid #F95959;  
 padding: 18px 36px;  
 border-radius: 30px;  
 font-weight: 400;  
 font-size: 16px;  
}

**Урок 12 - Компонент формы. Часть 1**

Секции Counseling и Отзывы пропустим (их необходимо реализовать самостоятельно).

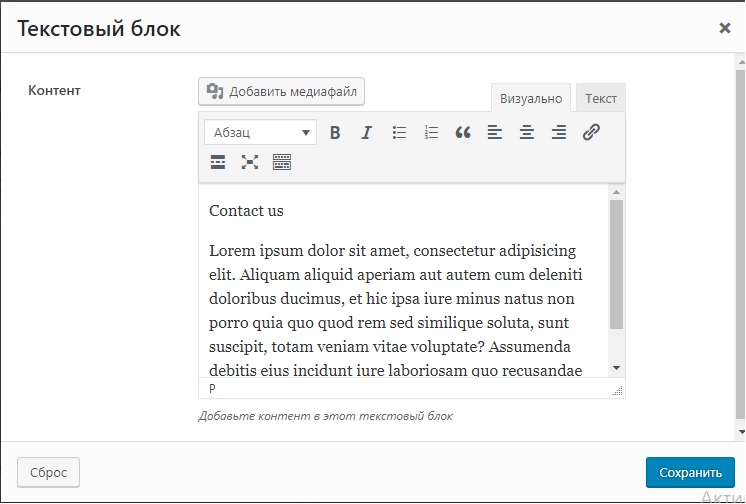
В данном уроке реализуем секцию с формой.

1е - Данная секция будет занимать всю ширину стр.

2е – в левой части вместо видео поместить просто текст, так как блок с видео ранее мы уже делали.

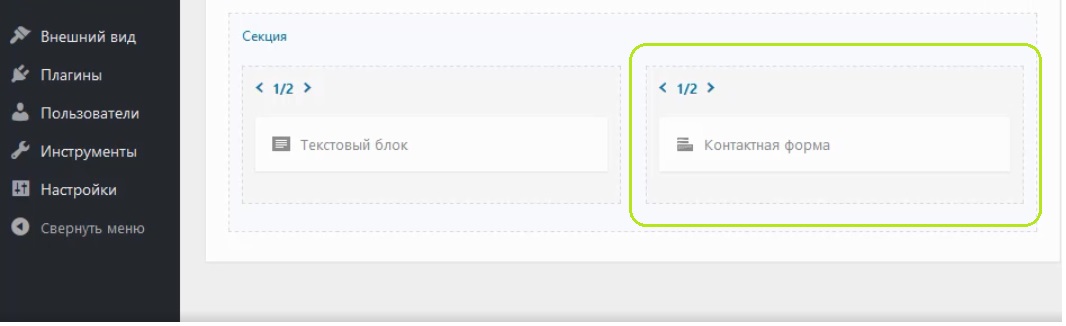
Как обычно создадим данную секцию в page builder:

Текстовый блок (левая часть):



Также колонке слева необходимо добавить классы ‘choose animate-box’ и еще можно сбросить отступы (в случае использ. видео), добавив класс текущей колонки в law.css

Далее сконструируем правую колонку:



Как альтернативный вариант можно использовать популярный плагин: **contact form 7** – позволяет генерировать поля форм и, что самое главное, **позволяет отправлять форму без перезагрузки страницы, асинхронно.**

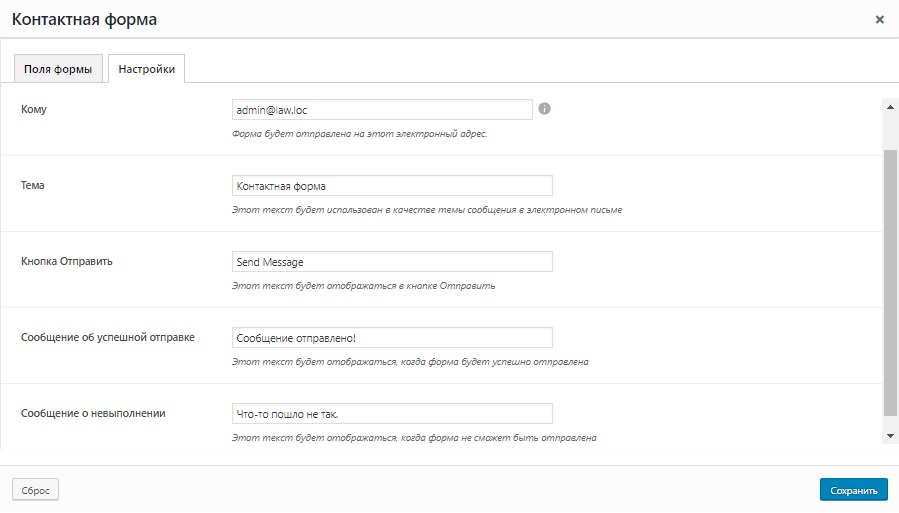
Когда мы создадим форму - CF7 сгенерит виджет который путем перетаскивания можно поместить в колонку или еще куда-либо.

Минус CF7 в том, что сгенерированную им форму сложно стилизовать.

**Виджет, идущий по умолч. в WP, конт.форма** стилизуется гораздо проще, но его минус в том, что он **работает синхронно**, т.е. **при отправке формы перезагрузится вся страница**.

В данном уроке воспольз. виджетом, идущим по умолч. в WP, конт.форма, но сделаем его асинхронным, так как синхронная форма это прошлый век.  
Настроим контакт. Форму в page builder:

*Семантически правильнее сделать кнопку при помощи тега <button>*



Затем указываем настройки мейлера, это способ отправки нашего письма:

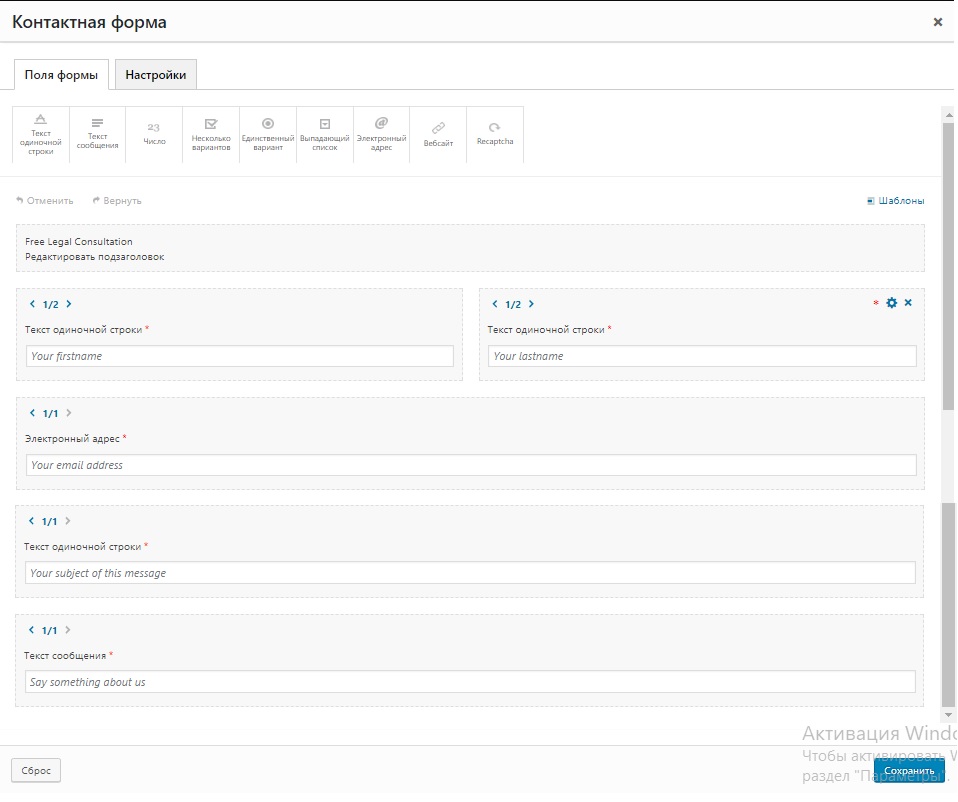


По умолч. это wp-mail.   
wp-mail – это функция обертка для функции из ядра php – mail().  
Т.е. письмо будет отправляться почтовой службой хостинга, но не всех хостингах предоставляется такая возможность, некоторые хостинги отключают возможность отправки писем, в этом случае можно воcпольз. сторонним SMTP, например, google SMTP сервер, в этом случае все настройки берутся на сайте SMTP сервера.

В учебных целях оставим wp-mail.

Для правой колонки указываем классы ‘choose animate-box’

Переходим к полям формы:



Кнопка подставляется автоматически.

Стилизовать форму предстоит самостоятельно.

Теперь необходимо стилизовать кнопку, сделать это можно несколькими путями при помощи js или немного поменять шаблон, мы пойдем вторым путем кастомизируем шаблон, копируем компонент /plugins/unyson/framework/extensions/forms/extensions/contact-forms

В framework-customizations /extensions/forms/extensions

Затем переписываем views/**submit.php** :

<div>  
 <button type="submit" class="btn btn-primary">  
 **<?php echo** esc\_attr( $submit\_button\_text ) **?>** </button>  
</div>

Если Fatal error: Cannot redeclare fw\_ext\_contact\_forms\_get\_option() (previously declared in C:\OSPanel\domains\law.loc\wp-content\plugins\unyson\framework\extensions\forms\extensions\contact-forms\helpers.php:13)

просто удаляем helpers.php, так как находящиеся в нем методы нельзя переопределить.

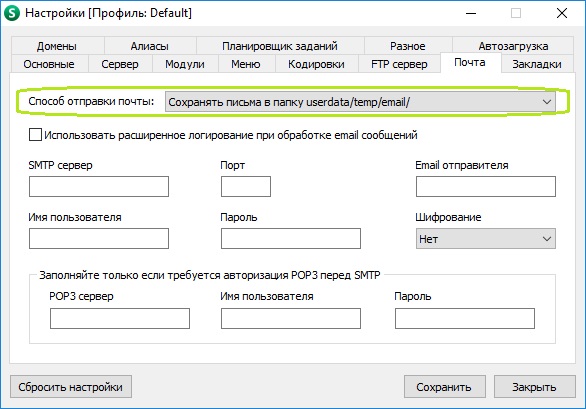
Затем добавим подсветку активному полю, копируем правило из исходного шаблона в law.css:

.fw\_form\_fw\_form .field-text input:active,  
.fw\_form\_fw\_form .field-text input:focus,  
.fw\_form\_fw\_form .field-textarea textarea:active,  
.fw\_form\_fw\_form .field-textarea textarea:focus {  
 outline: none;  
 box-shadow: none;  
 border-color: #F95959;  
}

Форма готова теперь её можно потестить, отправим письмо:

По умолчанию Open Server складывает письма в папку   
C:\OSPanel\userdata\temp\email

Если мы хотим, чтобы OS реально отправлял письма, необходимо настроить SMTP в settings OS:



Так как мы использ. встроенную возможность Unyson при отправке письма мы перенаправляемся в самый верх страницы, так как она перезагружается из-за синхронной работы формы. Плюс все стилизованно плохо, поэтому исправим эти моменты.

Продолжение в след. уроке.

**Урок 13 - Компонент формы. Часть 2**

В этом уроке сделаем нашу форму юзабельной.

Если обратить внимание на содержание письма, то мы увидим, что у значения нет подписи, не отмечено к какому полю относится значение

<table border="0" cellpadding="10">  
 <tbody>  
 <tr>  
 <td valign="top"><b></b></td>  
 <td valign="top">Alexander</td>  
 </tr>  
 <tr>  
 <td valign="top"><b></b></td>  
 <td valign="top">Ismailov</td>  
 </tr>  
 <tr>  
 <td valign="top"><b></b></td>  
 <td valign="top">smilesdcb@mail.ru</td>  
 </tr>  
 <tr>  
 <td valign="top"><b></b></td>  
 <td valign="top">Test mail</td>  
 </tr>  
 <tr>  
 <td valign="top"><b></b></td>  
 <td valign="top"><pre>Text of the test message</pre></td>  
 </tr>  
 </tbody>  
</table>

Чтобы это исправить необходимо в настройках контактных полей указать ярлык для каждого поля.

После этого на Frontend части появятся спец. отметки для обязательных полей. Чтобы их убрать из frontend части (если они не нравятся) необходимо добавить CSS правило:

.fw\_form\_fw\_form .wrap-forms label {  
 display: none;  
}

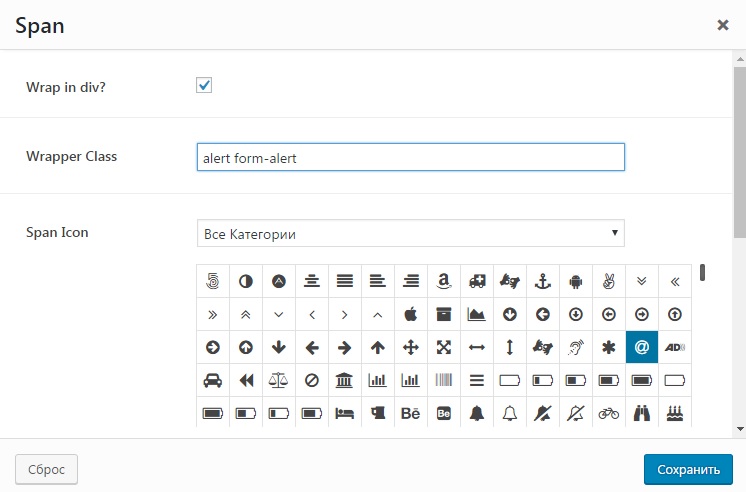
Но мы оставим его закомментированным, лишь сделаем ширину шрифта нормальной:

.fw\_form\_fw\_form .wrap-forms label {  
 /\*display: none;\*/  
 font-weight: 400;  
}

Теперь **настроим асинхронную отправку письма**:

Для этого потребуются возможности AJAX, в рамках уч. курса мы не будем создавать отдельный файл, а поместим наш ajax код в   
/assets/js/main.js

С начала в админке добавл. наш кастомный элемент Span, в ту же колонку, под контактную форму и настраиваем его:



В header добавим loader (крутящееся колесо), перед body пропишем:

<!--метка для Loader-->  
<div class="fh5co-loader"></div>

Этот span будет выводится в качестве информационного сообщения о результате отправки письма. В качестве шаблона возьмем инф.алерт из bootstrap 3 (так как исх. шаблон использ. 3ю версию). Если письмо отправленно удачно, то тогда spanu присваивается класс 'alert-success' в противном случае 'alert-danger'.

Также spanu необходимо будет указывать полный путь к гифке с колесом Loader.

Как узнать средствам JS путь к папке с темой и передать его в JS ?  
Для этого в клиентской части (на стороне frontend) создадим переменную куда поместим путь.  
Чтобы мы могли **динамически сформировать эту переменную**(путь к папке с темой, не зависимо от ее названия) в шаблоне (в functions.php) нам необходимо воспольз:

**wp\_localize\_script()** - <https://wp-kama.ru/function/wp_localize_script> - Добавляет дополнительные данные перед указанным скриптом, который должен быть в очереди на вывод.

Функцию wp\_localize\_script() надо поместить в functions.php сразу после скрипта, который будет использ. переданные ей данные.

. . . . .

. . . . .

wp\_enqueue\_script(  
 'law-main',  
 get\_template\_directory\_uri() . '/assets/js/main.js', **array**(), '', **true**);  
wp\_localize\_script( 'law-main', 'lawData', **array**(  
 'themePath' => get\_template\_directory\_uri()  
) );

. . . . .

. . . . .

Теперь мы имеем в клиенстой части объект-переменную lawData:

console.log(lawData);

>>> Object {themePath: "http://law.loc/wp-content/themes/law"}

Теперь допишем loader в main.js:

Так как url не указан ajax запрос будет обращен к текущей форме.

/\* Настройка асинхронной работы формы \*/  
 $('.fw\_form\_fw\_form').on('submit', **function**() {  
 **var** $this = $(**this**), // сама форма  
 btn = $this.find('button'), // сама кнопка для защиты от повторного нажатия во время отправки  
 data = $this.serialize(), // данные для отправки  
 formAlert = $('.form-alert'), // Элемент для вывода сообщения о результате отправки письма  
 loader = "<img class='loader' src='${lawData.themePath}/assets/images/loader.gif'>'" // крутящееся колесо во время загрузки

// Далее формируем AJAX запрос  
 $.ajax({  
 type: 'POST',  
 data: data,  
 beforesend: **function** () {  
 btn.attr('disabled', **true**);// блокируем кнопку  
 /\*  
 \* скроем не нужные классы перед повторной отправкой письма,  
 \* а также очистим саму форму alert  
 \* \*/  
 formAlert.removeClass('alert-success alert-danger').empty();  
 btn.after(loader);  
 },  
 success: **function** (responce) {

// при помощи fadeIn за 300мс выводим наш span

// и по окончанию вызываем callback function

formAlert.fadeIn( 300, **function** () {  
 console.log(responce);  
 } );  
 },  
 error: **function** () {  
 alert('Error');  
 }  
 });  
  
 // alert('It works');  
 // return false;  
});

в law.css на всякий случай добавляем:

.form-alert {  
 display: none;  
}

Теперь после отправки письма в консоле можно увидеть объект, который содержит результат отправки письма, и в случае успеха, содержание этого письма.

продолжение в следующем уроке....

**Урок 14 - Компонент формы. Часть 3**

Форма практически готова, она уже отправляется асинхронно. Осталось вывести сообщение о результате, разблокировать кнопку и убрать loader, для этого допишем метод success в ajax запросе:

$.ajax({  
 type: 'POST',  
 data: data,  
 beforeSend: **function** () {  
 btn.attr('disabled', **true**);// блокируем кнопку  
 /\*  
 \* скроем не нужные классы перед повторной отправкой письма,  
 \* а также очистим саму форму alert  
 \* \*/  
 formAlert.removeClass('alert-success alert-danger').empty();  
 btn.after(loader);  
 console.log(loader)  
 },  
 success: **function** ( responce ) {  
 // при помощи fadeIn за 300мс выводим наш span и по окончанию вызываем callback function  
 formAlert.fadeIn( 300, **function** () {  
 console.log( responce );// Test  
 **if**( responce.success ) {  
 formAlert.addClass( 'alert-success' ).text( responce.data.flash\_messages.success.fw\_ext\_contact\_form\_process );  
 $this[0].reset();// очистим поля формы  
 } **else** {  
 formAlert.addClass('alert-danger');  
 **var** errors = '';  
 **for**(**var** key **in** responce.data.errors) {  
 errors += responce.data.errors[key] + '<br>';  
 }  
 formAlert.html(errors);  
 }  
 formAlert.addClass();  
 } );  
  
 $('.loader').remove();  
 btn.attr('disabled', **false**);  
 },  
 error: **function** () {  
 alert('Error');  
 }  
 });  
  
 // alert('It works');  
 **return false**;  
});

**Урок 15 - Секция последних постов. Часть 1**

на странице front-page.php, сразу после стандартного цикла WP, выводящего контент, который мы навояли в Page builder (для статической страницы Home), организуем свой пользовательский цикл, который будет выводить последние добавленные записи.

Займемся выводом последних постов не посредственно в шаблоне front-page, создадим польз. запрос для вывода постов, укажем только один параметр, кол-во записей выводимых на странице:

**<?php** $query = **new** WP\_Query( **array**(  
 'posts\_per\_page' => 3 // кол-во записей на одной страничке  
 ) );  
**?>**

после этого переносим секцию из исх.шаблона

<div id="fh5co-blog" class="fh5co-bg-section"...>

во front-page.php сразу полсле WP\_Query, и оставляем только один

<div class="col-lg-4 col-md-4"...>

Этот div необходимо поместить в WP цикл:

<div id="fh5co-blog" class="fh5co-bg-section">  
 <div class="container">  
 <div class="row animate-box">  
 <div class="col-md-8 col-md-offset-2 text-center fh5co-heading">  
 <h2>Recent Post</h2>  
 <p>Dignissimos asperiores vitae velit veniam totam fuga molestias accusamus alias autem provident. Odit ab aliquam dolor eius.</p>  
 </div>  
 </div>  
 <div class="row">  
  
 **<?php if** ( $query->have\_posts() ) : **while**( $query->have\_posts() ) : $query->the\_post(); **?>** <div class="col-lg-4 col-md-4">  
 <div class="fh5co-blog animate-box">  
 <a href="**<?php** the\_permalink(); **?>**">**<?php** the\_post\_thumbnail(); **?>**</a>  
 <div class="blog-text">  
 <span class="posted\_on">Nov. 15th</span>  
 <span class="comment"><a href="">21<i class="icon-speech-bubble"></i></a></span>  
 <h3><a href="**<?php** the\_permalink(); **?>**">**<?php** the\_title(); **?>**</a></h3>  
 <p>**<?php** the\_content(''); **?>**</p>  
 <a href="**<?php** the\_permalink(); **?>**" class="btn btn-primary">**<?php** \_e('Read More,', 'law'); **?>**</a>  
 </div>  
 </div>  
 </div>  
 **<?php endwhile**; **?>** <!-- post navigation -->  
 **<?php else**: **?>** <!-- no posts found -->  
 **<?php endif**; **?>** </div>  
 </div>  
</div>

Далее добавим посты в админке.

Метки это теги по которым потом можно группировать статьи, метки можно вывести при помощи функции **the\_tags();**

Теперь выводится 3 записи в данной секции.

Немного допишем пользовательский цикл, добавим условие для вывода миниатюры к записи в карточке, поэтому удалем the\_post\_thumbnail() и вместо него указываем условие:

<a href="**<?php** the\_permalink(); **?>**">  
 **<?php  
 if** (has\_post\_thumbnail()) { // если лого есть, то использ. его адрес  
 $img\_url = get\_the\_post\_thumbnail\_url();  
 } **else** { // если лого нет, то картинка по умолч.  
 $img\_url = get\_template\_directory\_uri() . '/assets/images/moon.jpg';  
 }  
 **?>** <img src="**<?php echo** $img\_url; **?>**" alt="**<?php** the\_title(); **?>**">  
</a>

Далее выводим дату поста и кол-во комментариев к нему.

Что касается комментариев, то сейчас уже мало кто испольует встроенные WP комментарии, обычно прикручивают комменатрии из соц сетей.

За вывод даты отвечает

the\_time($d); - <https://wp-kama.ru/function/the_time> - $d(строка)

Формат даты (в PHP синтаксисе). Например, j F Y выведет "12 декабря 2010". Подробнее форматирование дат в PHP смотрите здесь: <https://wp-kama.ru/id_7433/formaty-daty-i-vremeni-v-wordpress.html>

За вывод только кол-ва комментариев отвечает **get\_comments\_number()**

перепишем блоки < span >

<span class="posted\_on">**<?php** the\_time('j M'); **?>**</span>  
<span class="comment"><a href="">**<?php echo** get\_comments\_number(); **?>**<i class="icon-speech-bubble"></i></a></span>

Поидее все готово, но есть одна трудность с которой может столкнуться пользователь, если он захочет поменять текст или описание в заголовке, или поменять кол-во выводимых записей, или если понадобится вообще скрыть данный шаблон, то придется лезть в шаблон проекта.

Избежать этого можно несколькими путями, например так как мы делали в CLEAN, где в кастомайзере указывали из какой категории выводить записи, и также там была галочка выводить ли вообще данную секцию с записями.

Но мы пойдем другим путем, создадим новый кастомный shortcode в Unyson, который назовем Recent Post, и собственно в этом компоненте мы и будем определять все выше перечисленные характеристики для данной секции:

кол-во записей

категория или категории из которой необходимо выводить записи (или если не выбрана, то категория по умолчанию)

продолжение в следующем уроке ....