Пояснительная записка к проекту

Блог путешественника «Choose travel».

Введение

Начну сначала. Почему я выбрал такую область для проекта?

Ранее, на первом курсе, нас “обучали” азам создания сайта (по сути обычному рисованию в Dreamweaver’е ), но на “выходе” уровень знаний в этой области остался прежним. Несмотря на это, тема: «верстка сайтов» все-таки заинтересовала меня. Но я не хотел заниматься обычным рисованием, как в Point’е, а хотел глубже изучить HTML и CSS. Поэтому, начал самостоятельно изучать их с помощью разных источников (благо, их в интернете немало). Изучение изучением, но какой-то большой работы у меня не было, были только теоретические знания, подкрепленные небольшими практическими заданиями.

Так, я, с периодичностью, изучал до середины-конца второго курса. И, когда пришло время выбирать тему проекта, у меня уже была идея сделать небольшой сайт, от и до. Чтобы уже на своем опыте познать все те нюансы, с которыми встречается разработчик сайтов. У меня было несколько критериев выбора сайта.

1. Он должен обладать кроссбраузерностью.

Аргументы по выбору этого критерия очень просты: сайт, который поддерживается только одним браузером просто не «выживет» на конкурентном рынке.

1. Он не должен быть очень большим/сложным.

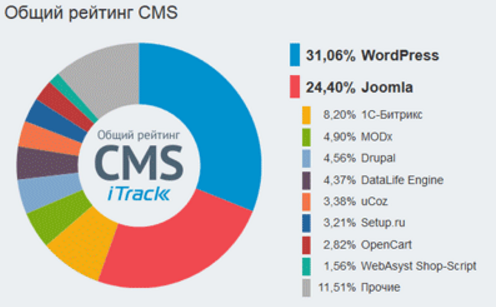
Большой/сложный сайт имеет большой риск быть недоделанным по нескольким причинам:

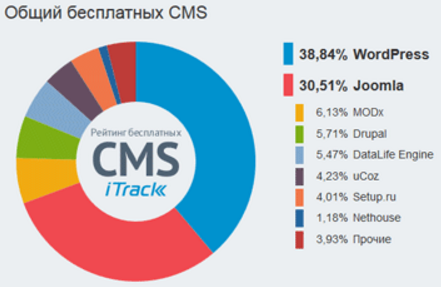
* Время на разработку проекта ограничено.
* Количество разработчиков проекта: 1.
* Неопытность.

1. Он должен быть на CMS’ке.

Я выбрал для этого проекта WordPress. Так как одной из своих целей проекта я ставил получить опыт в работе с Content Management System WordPress, а именно в создании собственной темы на WP. И так как он сейчас считается лидером на рынке CMS.

Вот несколько сведений о CMS’ках:





Исходя из этих всех и некоторых других целей, я составил техническое задание на свой проект.

**Техническое задание на создание сайта**

**“Choose travel”.**

1. Введение

Наименование проекта – Choose travel. Проект является несложным сайтом, который является блогом путешественника. Целью проекта является ознакомление и получение навыков в верстке, php, WP и создании темы на WP.

1. Основание для разработки

Работа ведется на основании документов:

- Технический план по предмету “Технологии разработки программного обеспечения”

- Персональные рекомендации для проекта по предмету “Технологии разработки программного обеспечения

1. Назначение разработки
   1. Функциональное назначение

Функциональным предназначением проекта является предоставление пользователю возможности ведения своего персонального блога путешественника.

* 1. Эксплуатационное назначение

Программа будет эксплуатироваться на любых браузерах.

3.2.1. Требование к функциональным характеристикам

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

* Функция добавления новой записи
* Функция добавления/удаления сайдбаров
* Функция комментирования, контроля комментариев
* Функция поиска по заданному(-ым) слову(-ав)
* Функция создания собственного меню
* Функция загрузки собственных изображений, как для слайда, так и для записей
* Функция составления собственного футера
* Функция установки количества записей на одной странице
* Функция отображения несуществующей страницы

3.2.2. Требования к надежности

Программа должна включать себя логику, обеспечивающую стабильную работу при нештатных ситуациях, то есть иметь механизм обработок ошибок. В случаях выявления отказов программы следует незамедлительно исправить возникшую проблему и впредь обеспечивать обработку подобных отказов.

3.2.3. Условия эксплуатации

Эксплуатация программы проходит в нормальном режиме при исправности технического оборудования. На исправность оборудование может повлиять износ его частей, воздействие повышенных температур, сильное электромагнитное излучение. Также ошибки могут появиться из-за неисправностей самой операционной системы либо браузера. Эксплуатация возможна, но затруднена при несоблюдении пользователем минимальных системных требований.

3.2.4. Детали реализации проекта

Задействованные языки программирования в данном проекте: HTML, CSS, PHP.

1. Требования к программной документации

В состав программной документации должны входить:

* Техническое задание

1. Технико-экономические показатели

Из-за относительно малого количества, простоты, но в то же время универсальности и достаточности функций в проекте можно сделать вывод, что программа будет являться экономически эффективной.

1. Стадии и этапы разработки

Разработка должна быть проведена в три стадии:

* Техническое задание
* Рекомендации по разработке
* Технический проект

1. Порядок контроля и приемки

Контроль работы над программой осуществляется на протяжении всего периода разработки, учитывается качество работы на каждом этапе. По окончанию разработки заказчик проводит итоговые испытания и проверку соответствия продукта необходимым критериям, после чего делает заключение и подписывает документ о выполнении работы.

1. Приложения к техническому заданию

Источники разработки:

* <https://wp-kama.ru/>
* <https://www.youtube.com/>
* Множество других сайтов по обучению HTML, CSS, PHP и работе в WordPress’е

Во время всего создания данного проекта я опирался на это тз.

В следующем разделе, «Реализация», я опишу как я создавал тему на WordPress.

Реализация

В каждой теме на WordPress существует определенная структура файлов. Все файлы хранятся в папке с названием темы. Вот так:



WordPress пытается показать контент в наиболее подходящем для этого шаблоне. Если такого шаблона не будет, то WordPress обратится к выше стоящему по иерархии шаблону, если и такого не найдется, то он отобразится в «index.php».

Самые важные шаблоны, которые должны быть в любой теме – это «index.php» и «style.css». «index.php» отвечает за вывод главной страницы сайта, а «style.css» отвечает за стили сайта (внешний вид сайта), в том числе и для главной страницы. Если этих двух файлов не будет, то тема не заработает.

В свое время, «index.php» может разбиться на составляющие, такие как «header.php», «sidebar.php», «footer.php» и пр. Это удобно сделать, если последующие страницы будут содержать эти составляющие, и, чтобы постоянно не копировать многострочный код, достаточно будет написать <?php get\_header(); ?> (в данном случае подключили «header.php»). Для наглядности, приведу пример из своей темы файла «index.php»:

<?php get\_header(); ?>

<div class="content">

            <?php if (have\_posts()) : while (have\_posts()) : the\_post(); ?>

                <div class="post-main">

    <h1><a href="<?php the\_permalink(); ?>"><?php the\_title(); ?></a><span> <?php the\_time( 'j.m.Y' ) ?></span></h1>

    <div class="post">

    <?php the\_post\_thumbnail('full', 'class=imgstyle'); ?>

         <?php the\_excerpt(); ?>

         <p><a href="<?php the\_permalink(); ?>">Читать далее</a></p>

         <p><?php the\_tags(); ?> </p>

    </div>

    </div>

            <?php endwhile; ?>

<div class="pagination">

<?php

global $wp\_query;

$big = 999999999;

echo paginate\_links( array(

'base' => str\_replace($big, '%#%', esc\_url( get\_pagenum\_link( $big ) )),

'format' => '?paged=%#%',

'current' => max(1, get\_query\_var( 'paged' ) ),

'total' => $wp\_query->max\_num\_pages,

'prev\_text' => '&laquo;',

'next\_text' => '&raquo;'

) );

?>

</div>

            <?php else: ?>

            <?php endif; ?>

</div>

<?php get\_sidebar( ); ?>

</div>

<?php get\_footer(); ?>

Выделенные желтым цветом – это подключение отдельных файлов («header.php», «sidebar.php», «footer.php»), которые хранятся в той же директории, что и «index.php».

А вот и сам «header.php»:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title> <?php bloginfo('name'); ?>;

wp\_title(); </title>

<?php wp\_head(); ?>

</head>

<body>

<div class="main">

<div class="head">

<h2>Выбери своё<br/>Путешествие</h2>

<a href="<?php echo home\_url(); ?>"><span>C</span>hoose <span>T</span>ravel</a>

</div>

<div class="content-main">

<?php wp\_nav\_menu(array('theme\_location' => 'menu', 'container' => false)); ?>

<form class="search-main" action="<?php echo home\_url( '/' ); ?>" method="">

<input class="search-txt" type="text" name="s">

<input class="search-btn" type="image" src="<?php bloginfo('template\_url') ?>/images/serach-btn.jpg">

</form>

<div id="slider">

<ul id="cycle">

<li><img src="<?php bloginfo('template\_url') ?>/images/galery.jpg" height="266" width="927" alt="Slides" /></li>

<li><img src="<?php bloginfo('template\_url') ?>/images/kolizey.jpg" height="266" width="927" alt="Slides" /></li>

<li><img src="<?php bloginfo('template\_url') ?>/images/piramide.jpg" height="266" width="927" alt="Slides" /></li>

<li><img src="<?php bloginfo('template\_url') ?>/images/galery.jpg" height="266" width="927" alt="Slides" /></li>

<li><img src="<?php bloginfo('template\_url') ?>/images/kolizey.jpg" height="266" width="927" alt="Slides" /></li>

<li><img src="<?php bloginfo('template\_url') ?>/images/piramide.jpg" height="266" width="927" alt="Slides" /></li>

</ul>

<div id="cyclePager"></div>

     </div>

Если с главной страницей нашего сайта стало все ясно, то что делать с другими, например, в моем случае это страницы отдельных статей?

Как я уже в начале уже говорил, если WordPress не находит подходящий файлов для отображения той или иной страницы, то он отображает контент с помощью главной страницы, то есть «index.php».

Для вывода контента отдельной статьи используются несколько шаблонов, но чаще всего это «single.php».

У меня файл «single.php» отличается от главной страницы незначительно.

1. Я убрал блок, отвечающий за пагинацию:

<div class="pagination">

<?php

global $wp\_query;

$big = 999999999;

echo paginate\_links( array(

'base' => str\_replace($big, '%#%', esc\_url( get\_pagenum\_link( $big ) )),

'format' => '?paged=%#%',

'current' => max(1, get\_query\_var( 'paged' ) ),

'total' => $wp\_query->max\_num\_pages,

'prev\_text' => '&laquo;',

'next\_text' => '&raquo;'

) );

?>

</div>

1. Вместо названия статьи-ссылки

<h1><a href="<?php the\_permalink(); ?>"><?php the\_title(); ?></a><span> <?php the\_time( 'j.m.Y' ) ?></span></h1>

оставил просто название

<h1><?php the\_title(); ?><span> <?php the\_time( 'j.m.Y' ) ?></span></h1>

1. вместо вывода цитаты <?php the\_excerpt(); ?>

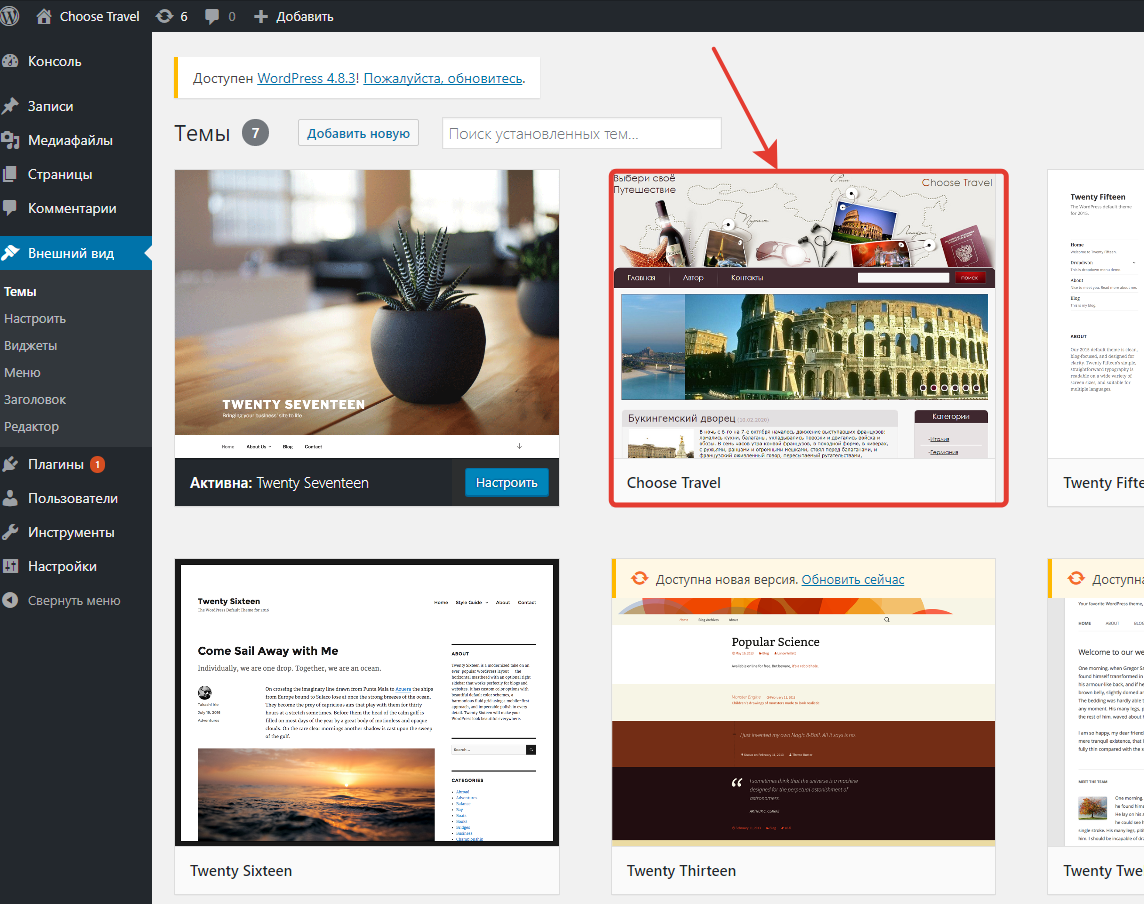
выводится полностью текст <?php the\_content(); ?>

1. Добавил вывод комментариев по этой статье

<?php comments\_template(); ?>

Для каких-то других страниц, например, для страницы, отображающей информацию об авторе, есть еще один специальный шаблон «page.php». Его мы тоже настраиваем так, как нами удобно. За основу можно взять либо «index.php», либо «single.php».

Чуть вернемся назад, и вспомним, что при создании темы в WP уже были готовые несколько тем («twenty seventeen», «twenty twelve» и др.), дак вот, у них в ячейке еще есть и фото, характеризующее тему. Чтобы такая картинка, появилась и у нас, нам надо загрузить фотографию, и дать ей название «screenshot». WP, видя картинку с таким названием, автоматически ставит ее в описание темы. Вот сюда:



Теперь снова возвращаемся к основным файлам нашей темы.

Сейчас мы рассмотрим файл «function.php».

Для безопасной постановки всех файлов стилей и всех скриптов в очередь на загрузку документация WP рекомендует использовать отдельный файл «function.php». Так же, исходя из названия, можно понять, что еще в этом файле записываются и php функции. Более подробно про этот файл можно прочитать [статью](https://wp-kama.ru/id_71/fayl-shablona-functionsphp-v-wordpress.html), которая в свое время мне помогла разобраться с этим файлом.

Здесь я подключил js файлы и css файл

/\*\*

\*загружаемые стили и скрипты

\*\*/

function load\_style\_script(){

wp\_enqueue\_script('jquery-1',get\_template\_directory\_uri() . '/js/jquery-1.js');

wp\_enqueue\_script('jquery00',get\_template\_directory\_uri() . '/js/jquery00.js');

wp\_enqueue\_script('init0000',get\_template\_directory\_uri() . '/js/init0000.js');

    wp\_enqueue\_style('style', get\_template\_directory\_uri() . '/style.css');

}

/\*\*

\*загружаем стили с скрипты

\*\*/

add\_action('wp\_enqueue\_scripts','load\_style\_script');

Еще один файл «category.php», он используется для показа контента рубрики. Я его сделал также, на основе «index.php».

Но для вывода названия рубрики вместо <?php the\_title(); ?>, в «category.php» используется

<?php single\_cat\_title("Рубрика:" ); ?>

Еще один шаблон – «search.php», из названия понятно, что он используется для вывода страниц по поиску. При отсутствии постов (не найден ни один пост) мы в конструкцию (помеченную желтым) добавляем строчку (помеченную зеленым).

            <?php if ( have\_posts() ) : while ( have\_posts() ) : the\_post(); ?>

                <div class="post-main">

    <h1><a href="<?php the\_permalink(); ?>"><?php the\_title(); ?></a><span> <?php the\_time( 'j.m.Y' ) ?></span></h1>

    <div class="post">

    <?php the\_post\_thumbnail('full', 'class=imgstyle'); ?>

         <?php the\_excerpt(); ?>

         <p><a href="<?php the\_permalink(); ?>">Читать далее</a></p>

    </div>

    </div>

            <?php endwhile; ?>

<div class="pagination">

<?php

global $wp\_query;

$big = 999999999;

echo paginate\_links( array(

'base' => str\_replace($big, '%#%', esc\_url( get\_pagenum\_link( $big ) )),

'format' => '?paged=%#%',

'current' => max(1, get\_query\_var( 'paged' ) ),

'total' => $wp\_query->max\_num\_pages,

'prev\_text' => '&laquo;',

'next\_text' => '&raquo;'

) );

?>

</div>

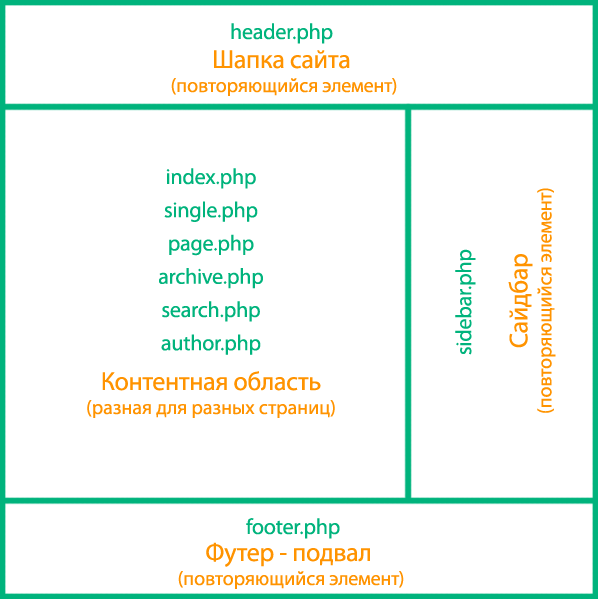
            <?php else: ?>

<p> Ничего не найдено :( </p>

            <?php endif; ?>

Еще есть файл «404.php» - он отображает не найденную страницу.

Для более четкого представления о структуре файлов предлагаю рассмотреть следующие картинки:





Подведу небольшой итог по данному разделу.

Чтобы создать свою тему на WordPress нам нужны файлы с определенными названиями. В этих файлах мы можем использовать специальные теги WordPress.

Небольшой пример таких тегов:

**the\_time('M'), the\_time('j') -**вывод месяца и дня соответственно  
**the\_permalink()** - отображает постоянную ссылку записи   
**the\_title\_attribute()**- заголовок текущей записи.  
**the\_title()**- заголовок записи или страницы   
**the\_category(', ')** - ссылку на категорию или категории, к которым принадлежит запись   
**comments\_popup\_link -**выведет ссылку для комментариев во всплывающем окне если comments\_ popup\_script() используется, иначе выводит обычную ссылку на комментарии.  
**the\_content** - выводит содержание текущей записи. **single\_cat\_title** - заголовок записи или страницы для шаблона категорий.

Все эти теги должны быть заключены в тег <?php ?>.

Заключение

В заключение хочу сказать. Что этот проект мне дал понимание работы WP. Дал базовые знания по PHP, js. А так же, я получил хороший опыт в верстке.

Но, конечно, есть и недочеты. В следующих своих работах я шире буду использовать возможности семантической разметки. Буду шире использовать возможности css для более интересного интерфейса.

Я считаю, что только опытным путем, делая разные работы, проекты, я смогу увеличивать свои знания и опыт. И конечно будут присутствовать ошибки, как раз они и заставляют нас идти дальше и развиваться.

По поводу WordPress могу сказать, что мне понравилось работать с ним, и я в будущем обязательно еще буду к нему обращаться.