Департамент образования, науки и молодежной политики
Воронежской области

Государственное бюджетное профессиональное образовательное учреждение
Воронежской области
«Воронежский государственный профессионально-педагогический колледж»

Цикловая методическая комиссия профессионального цикла специальности
компьютерные системы и комплексы, математических и естественнонаучных
дисциплин

С.В. Дорохов, М.В. Дорохова

Программное обеспечение компьютерных сетей и Web-серверов

Практикум

Воронеж, 2019

Департамент образования, науки и молодежной политики
Воронежской области

Государственное образовательное бюджетное учреждение
среднего профессионального образования Воронежской области
«Воронежский государственный профессионально-педагогический колледж»

Цикловая методическая комиссия профессионального цикла специальности
компьютерные системы и комплексы, математических и естественнонаучных
дисциплин

С.В. Дорохов, М.В. Дорохова

Программное обеспечение компьютерных сетей и Web-серверов

Практикум

Рекомендовано
советом учебно-методического центра
в качестве учебно-методического пособия по дисциплине
«Программное обеспечение компьютерных сетей и Web-серверов»
для студентов колледжа
специальности 44.02.06 «Профессиональное обучение» (по отраслям),
специальности профиля подготовки Компьютерные системы и комплексы

Воронеж, 2019

УДК 004.4

ББК 32.972.5

Д 69

Рецензенты:

Малева А.А., кандидат педагогически наук, доцент кафедры информатики и методики преподавания математики ФГБОУ ВПО «Воронежский государственный педагогический университет».

Савченко Е.А., председатель предметно-цикловой комиссии математических дисциплин и информационных технологий ГБПОУ ВО «Воронежский государственный профессионально педагогический колледж».

Дорохов С.В., Дорохова М.В., Программное обеспечение компьютерных сетей и Web-серверов: практикум – Воронеж: ВГППК, 2019. - 111 с.

Практикум соответствует рабочей программе по дисциплине «Программное обеспечение компьютерных сетей и Web-серверов» ФГОС 3+ СПО специальности Компьютерные системы и комплексы.

Учебное пособие призвано ознакомить студентов с основными приемами работы с системой управления контентом WordPress, построения собственных тем для системы управления контентом WordPress на основе имеющегося свёрстанного макета сайта.

Представленный материал служит справочным и методическим пособием при выполнении курса практических работ по дисциплине «Программное обеспечение компьютерных сетей и Web-серверов».

Представленный практикум может быть использован студентами средних специальных учебных заведений, изучающих дисциплину «Программное обеспечение компьютерных сетей и Web-серверов».

Ил. 112. Библиограф.: 6 назв.

Рассмотрено на заседании цикловой методической комиссии профессионального цикла специальности компьютерные системы и комплексы, математических и естественнонаучных дисциплин (Протокол №8 от 9.04.2019 г.)

Печатается по решению совета учебно-методического центра ГБПОУ ВО «Воронежский государственный профессионально-педагогический колледж» (протокол №5 от 20.05.2019 г.)

Содержание

Пояснительная записка

Цель методических указаний: оказание помощи студентам в выполнении практических работ по дисциплине «Программное обеспечение компьютерных сетей и Web-серверов».

Настоящий практикум содержит рекомендации к выполнению практических работ, которые позволят студентам закрепить полученные теоретический знания по дисциплине «Программное обеспечение компьютерных сетей и Web-серверов» курса дисциплины и направлены на формирование следующих компетенций:

ОК 1 Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес.

ОК 2 Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.

ОК 3 Принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность.

ОК 4 Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.

ОК 5 Использовать информационно-коммуникационные технологии в профессиональной деятельности.

ОК 6 Работать в коллективе и команде, эффективно общаться с коллегами, руководством, потребителями.

ОК 7 Брать на себя ответственность за работу членов команды (подчиненных), результат выполнения заданий.

ОК 8 Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации.

ОК 9 Ориентироваться в условиях частой смены технологий в профессиональной деятельности

ПК 3.1 Проводить контроль, диагностику и восстановление работоспособности компьютерных систем и комплексов.

ПК 3.2 Проводить системотехническое обслуживание компьютерных систем и комплексов.

ПК 3.3 Принимать участие в отладке и технических испытаниях компьютерных систем и комплексов; инсталляции, конфигурировании программного обеспечения.

В результате выполнения практических работ по дисциплине «Программное обеспечение компьютерных сетей и Web-серверов» студенты должны:

уметь:

- разбираться в коде разметки гипертекста и стилизации элементов web-страниц, написанных другим разработчиком;
- производить разбиение на структурные элементы файлы разметки гипертекста;
- определять элементы сайта требующие автоматизации обработки при помощи серверных и клиентских скриптов, либо средствами функций CMS;
- находить элементы, требующие дополнительной стилизации или пере стилизации в соответствии с макетом верстки или графическим представлением;
- определять для себя правильную последовательность работы над проектом.

знать:

- язык разметки гипертекста;
- основы стилизации элементов web-страниц при помощи каскадных таблиц стилей;
- языки серверных и клиентских скриптов;
- основы работы с системами управления контентом;

владеть:

- основными навыками и приемами работы над разработкой web-приложений;
- навыками работы по разработке темы оформления для CMS WordPress на основе макета верстки собственной или сторонней разработки.

Описание каждой практической работы содержит: тему, цели работы, оборудование и материалы, методические указания, порядок выполнения работы, задания для самостоятельной работы над собственным проектом, а также перечень контрольных вопросов.

Практическая работа № 1

Установка и первоначальная настройка CMS WordPress

Цели работы:

- Освоить основные приемы работы с CMS WordPress
- Научиться изменять основные параметры системы, изменять визуальный редактор;
- Научиться создавать статьи, рубрики и записи для публикации на сайте.

Оборудование и материалы:

- Персональный компьютер с установленной операционной системой семейства Windows;
- Локальный web-сервер OpenServer;
- Дистрибутив системы управления контентом WordPress.

Указания к выполнению

1.1 Установка CMS WordPress

Для установки CMS WordPress как на локальный сервер, так и на хостинг, его необходимо скачать. Перейдите к сайту расположенному по адресу https://ru.wordpress.org/, и нажмите кнопку «Получить WordPress», а в открывшемся новом окне кнопку «Скачать WordPress» с указанием текущей версии CMS. Также последнюю русифицированную версию всегда можно скачать по ссылке https://ru.wordpress.org/latest-ru_RU.zip.

Для работы, рассматриваемой CMS, необходимо соблюдение следующих минимальных системных требований:

- PHP версии 5.2.4 или выше (рекомендуется 5.6 и выше);
- MySQL версии 5.6 или выше (рекомендуется 5.6).

Как и любой другой PHP CMS, CMS Wordpress также требуется собственная база данных, её мы можем создать при помощи инструмента «PhpMyAdmin», входящего в состав любого из версий локального web-сервера OpenServer. (Во избежание потери данных расположенных на учебных

компьютерах, создайте на диске D, свою рабочую папку (Лучше если с использованием в наименовании только латинских символов, символов подчеркивания и цифр)), а уже в неё распакуйте дистрибутив OpenServer (на сетевом диске присутствует актуальная версия дистрибутива для всех версий).

После распаковки дистрибутива настройте OpenServer согласно системным требованиям и запустите его (*В случае наличия ошибок при запуске OpenServer обратитесь к за помощью преподавателя*).

Откройте инструмент PhpMyAdmin и авторизуйтесь в нём (*В системном трее необходимо щелкнуть любой из клавиш мыши, выбрать в всплывающем меню пункт «Дополнительно», а в нём пункт «PhpMyAdmin». Пользователь root, пароль по умолчанию отсутствует*). Перейдите к меню, расположенному слева и нажмите кнопку «Создать БД». В открывшемся окне «Имя базы данных», введите наименование создаваемой базы, например, wpdb и нажмите кнопку «Создать». Будет создана база данных с требуемым именем, а наша работа с инструментом PhpMyAdmin на данном этапе завершена Вы можете его закрыть.

В расположении ранее подготовленного локального web-сервера OpenServer, присутствует папка «domains», создайте в ней папку WordPress и произведите извлечение в её корень всех файлов ранее скачанного дистрибутива CMS WordPress. Перезагрузите OpenServer, после чего в списке сайтов вашего сервера появится сайт с доменным именем WordPress, откройте его (*меню OpenServer – Мои сайты - WordPress*). Пример расположения данных пунктов меню OpenServer изображен на рисунке 1.1.

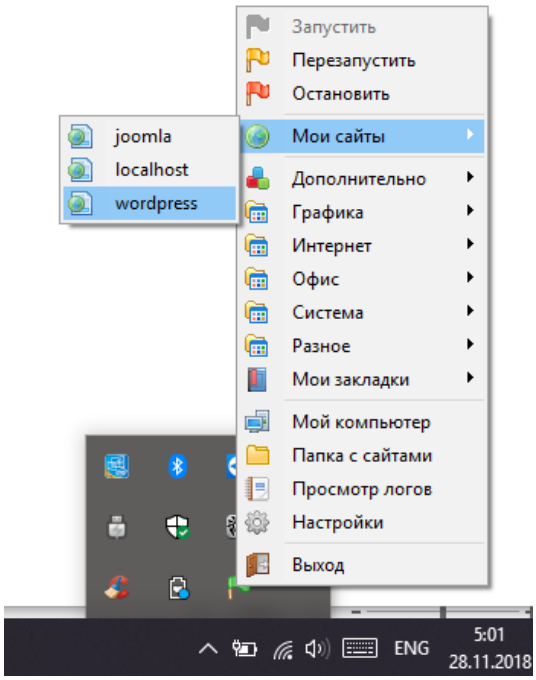


Рисунок 1.1 – Пример открытия сайта в панели управления OpenServer

Перед Вами откроется стартовое окно установки CMS WordPress изображенное на рисунке 1.2.

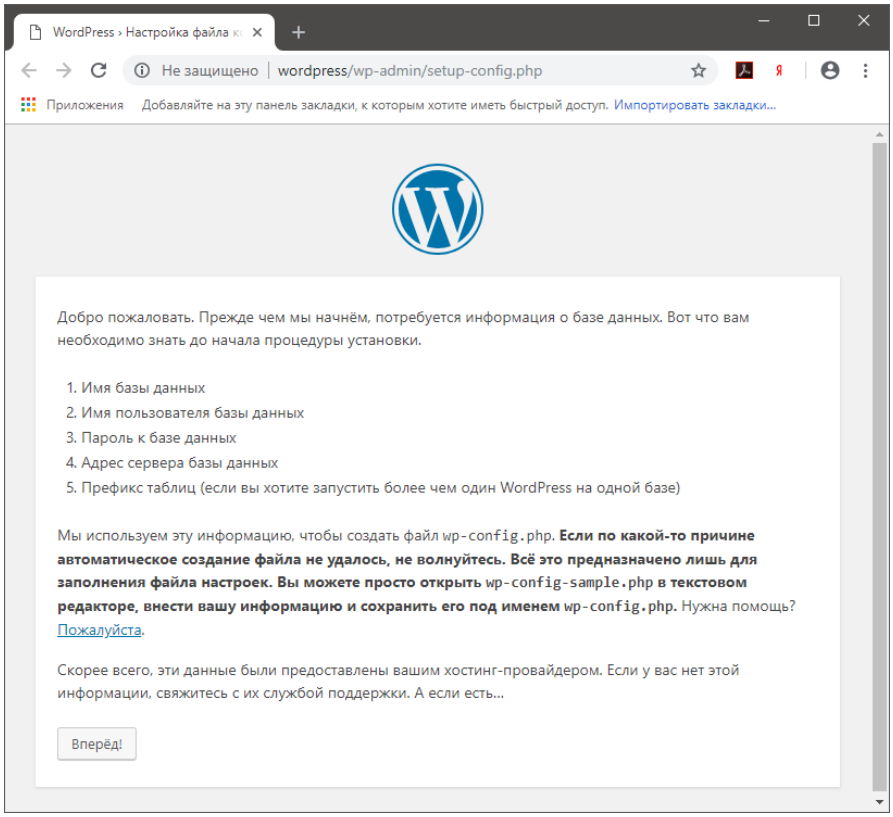


Рисунок 1.2 – Стартовое окно установки CMS WordPress

В данном окне будет представлена информация о данных, необходимых для установки и формирования конфигурационного файла.

Обратите внимание на то, что возможно использование так называемого префикса таблиц базы данных. Данная возможность полезна при размещении сайта на хостинговых площадках, когда существует ограничение по количеству используемых баз данных. Используя её, возможно хранение баз данных различных сайтов в одной базе, имеющей различные префиксы (начальные символы базы данных каждого из сайтов).

После ознакомления с имеющейся информацией, нажмите кнопку «Вперед».

В появившемся окне конфигурации подключения к базе данных введите следующие данные:

- Имя базы данных – wpdb;
- Имя пользователя – root;
- Пароль – оставьте поле пустым (очистите имеющееся значение по умолчанию);
- Сервер базы данных – localhost;
- Префикс таблиц – оставьте значение по умолчанию (wp_).

Внешний вид окна конфигурации базы данных с заполненными значениями приведен на рисунке 1.3. После введения данных о базе данных нажмите кнопку «Отправить».

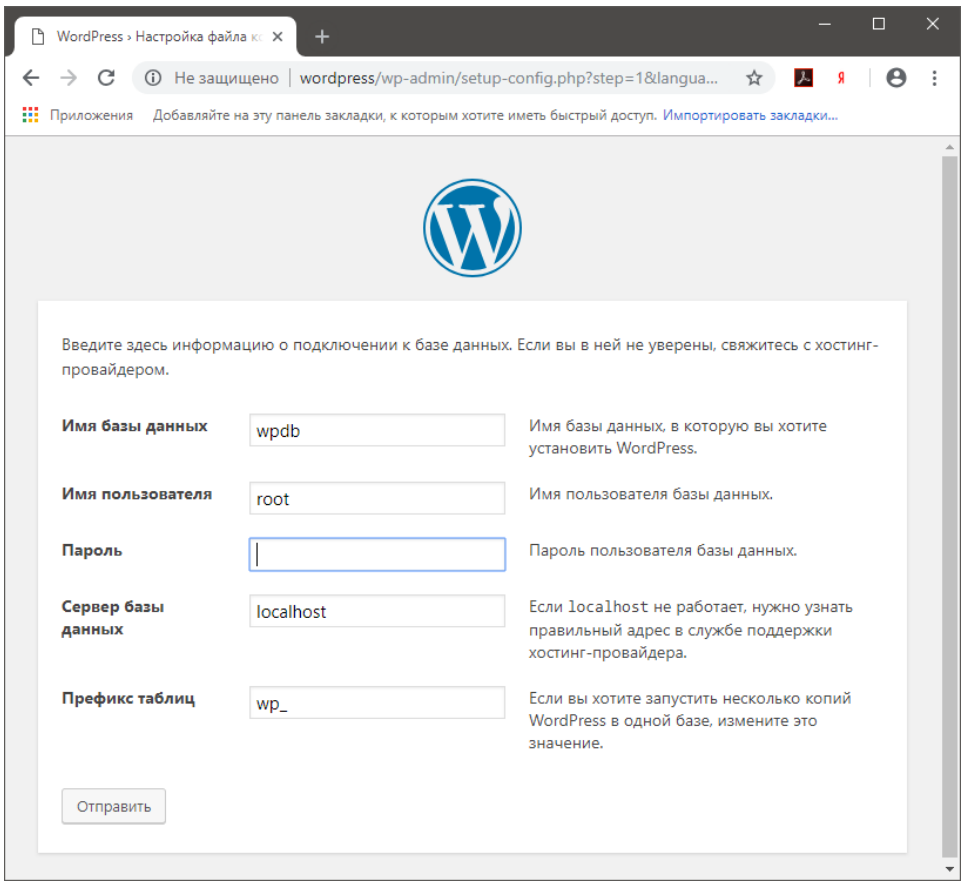


Рисунок 1.3 – Окно установки CMS WordPress, при конфигурировании подключения к базе данных

В случае безошибочной конфигурации и успешного прохождения теста подключения, перед Вами откроется окно программы установки, изображенное на рисунке 1.4, нажмите в нем кнопку «Запустить установку».

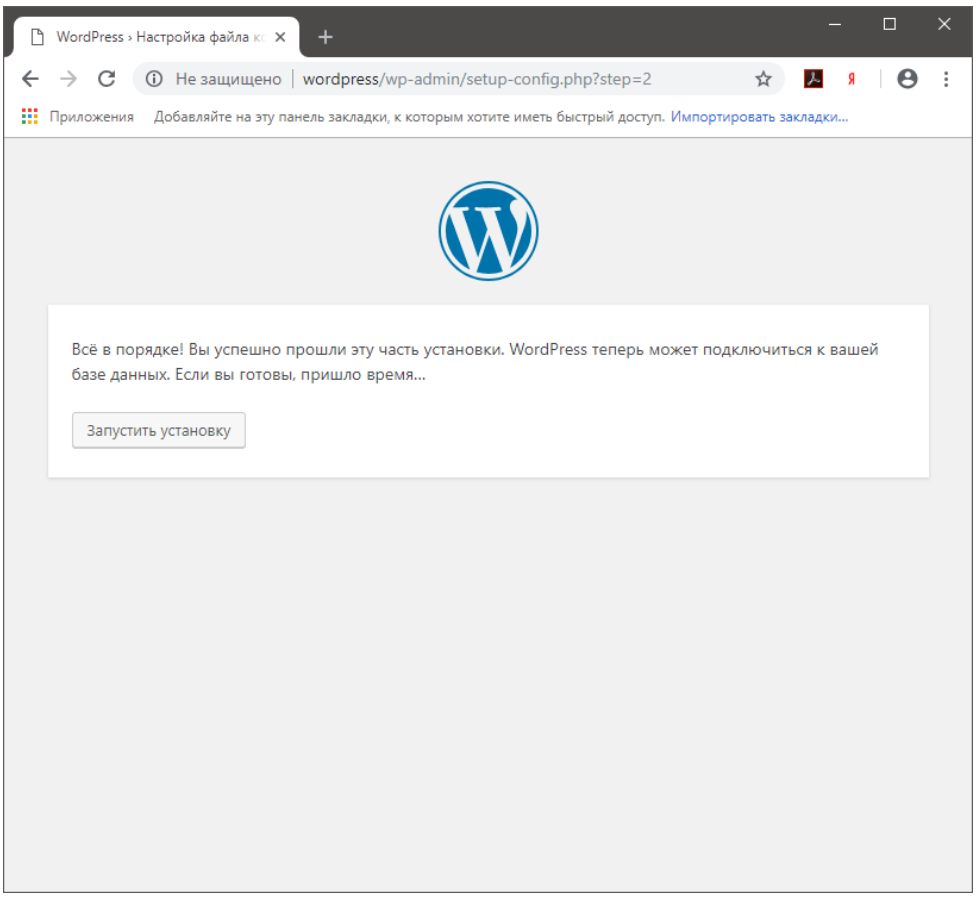


Рисунок 1.4 – Окно успешной проверки конфигурации подключения к базе данных, при установке CMS WordPress

Перед Вами откроется финальное окно установки CMS WordPress. Заполните его поля следующей информацией:

- Название сайта – Keep It Simple;
- Имя пользователя – admin;
- Пароль – 123qwe!@#;
- Ваш e-mail – введите Ваш собственный адрес e-mail, возможно в дальнейшем Вам будет необходимо воспользоваться функцией восстановления доступа к администрированию создаваемого сайта;
- Не забудьте отметить пункты: «Разрешить использование слабого пароля» и «Попросить поисковые системы не индексировать сайт».

Окно программы установки с примером внесенных данных изображено на рисунке 1.5.

После внесения всех данных, нажмите кнопку «Установить WordPress», после чего начнется непосредственно установка CMS.

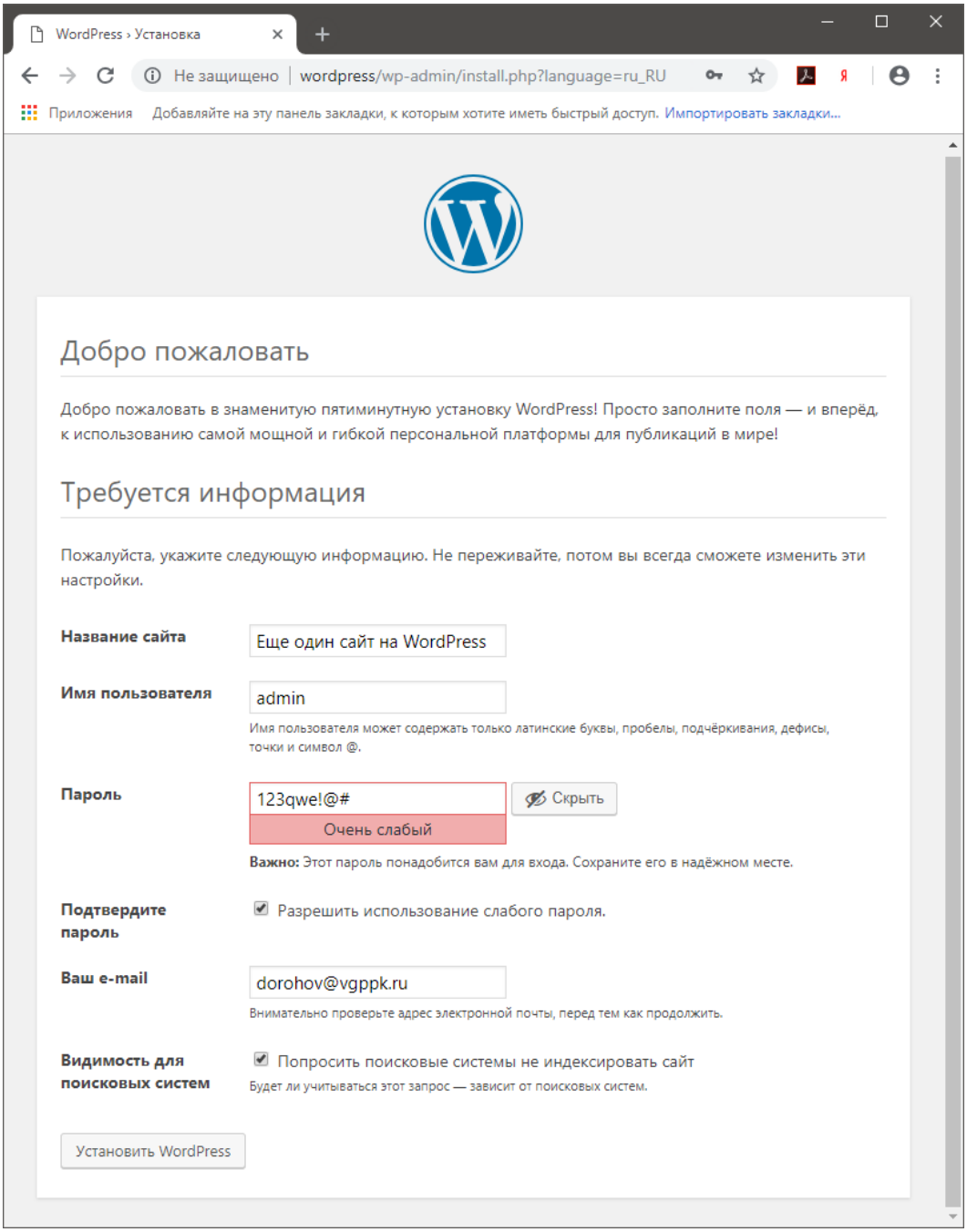


Рисунок 1.5 – Окно конфигурирования наименования сайта и доступа к администрированию CMS WordPress

По окончании установки, перед Вами откроется окно, сообщающее о успешной установке системы с возможностью перехода к панели авторизации пользователя (Рисунок 1.6) в консоли управления сайтом изображённое на рисунке 1.7.

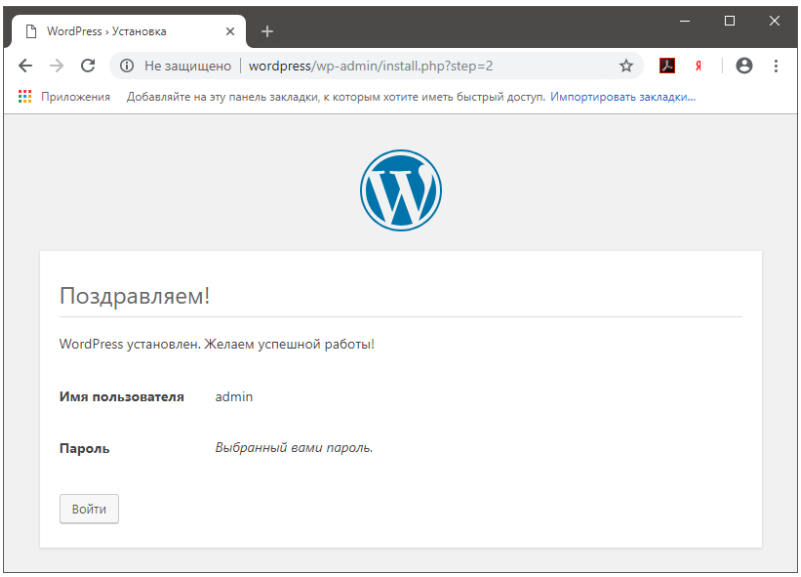


Рисунок 1.6 – Окно успешной установки CMS WordPress

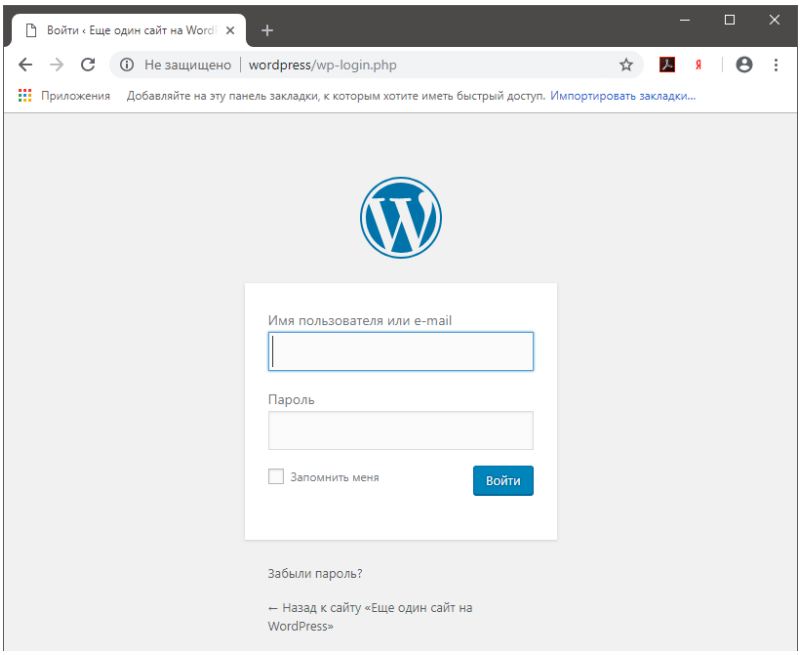


Рисунок 1.7 – Окно авторизации в консоли управления CMS WordPress

На этом, установка CMS WordPress завершена.

В дальнейшем для перехода к панели управления будет использоваться адрес http://wordpress/wp-login.php, а сам сайт будет расположен по адресу http://wordpress.

1.2 Первоначальная настройка CMS WordPress

После установки CMS, перейдите к окну авторизации в консоли администратора по адресу http://wordpress/wp-login.php и используя свои учетные данные, ранее введенные при установке, пройдите авторизацию. После авторизации перед Вами откроется окно консоли администратора CMS WordPress изображенное на рисунке 1.8.

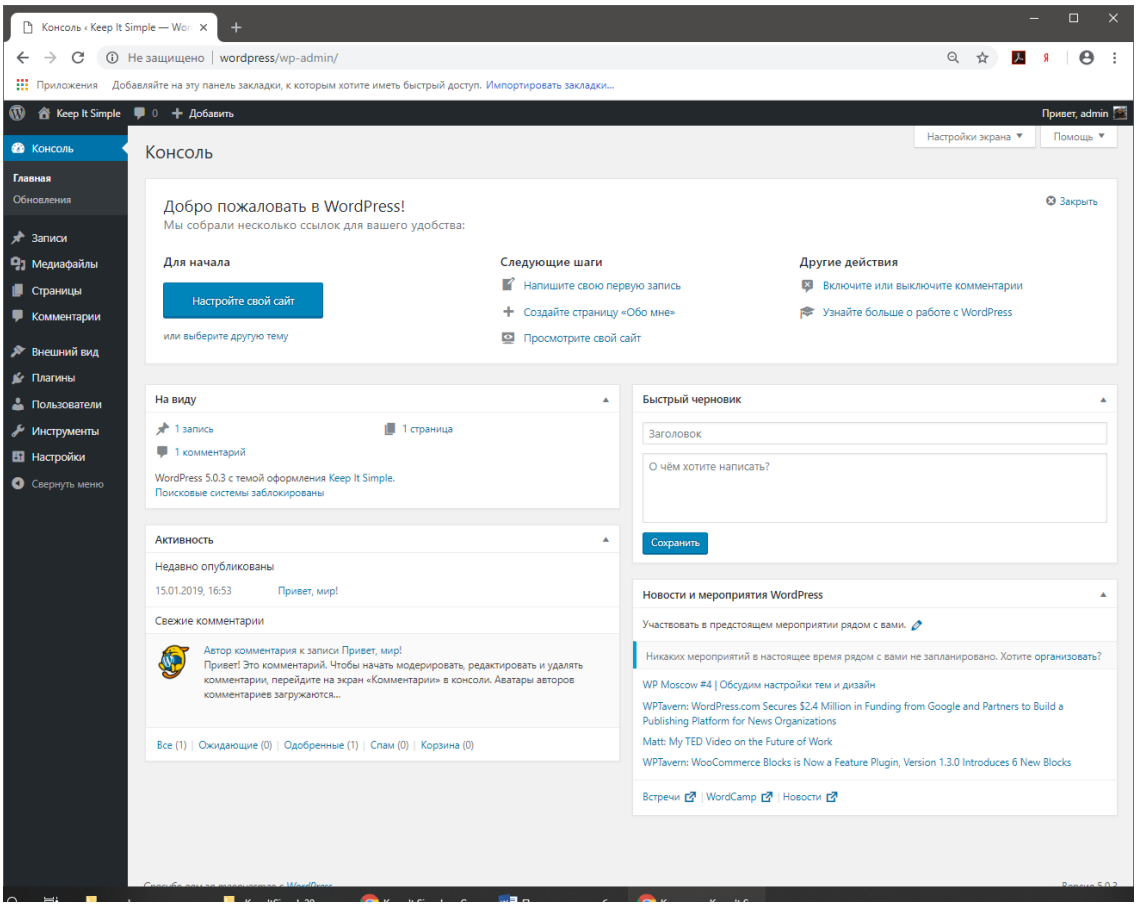


Рисунок 1.8 – Начальная страница консоли администратора CMS WordPress

Консоль WordPress имеет гибкий и настраиваемый интерфейс. Просмотрите отображение сайта (*для этого необходимо нажать на название сайта в верхнем левом углу консоли администратора*).

В верхней части открытой страницы сайта, Вы можете видеть черную полосу, эта полоса является рабочей панелью консоли управления сайта на WordPress, и отображается она только в том случае если Вы авторизовались в консоли управления. Одним из основных преимуществ этой панели, является возможность лёгкого перехода от сайта к консоли и обратно, для этого

необходимо кликнуть по наименованию сайта в левом верхнем углу сайта, после чего вы вернетесь обратно в консоль. Таким же образом, Вы можете вернуться к сайту.

Первым системным сообщением является предложение произвести первоначальные настройки Вашего сайта. Нажмите в данном окне на кнопку «Настройте свой сайт». Перед Вами откроется окно настроек сайта (*для текущей темы оформления*), набор возможностей для настроек зависит от функционала темы. Изучите пункты предоставленного меню, отредактируйте значения параметром по своему вкусу, например, цветовую схему. Обратите внимание, что при этом происходит «живое» отображение изменений на сайте. Также можно изменить название сайта и его краткое описание, добавить логотип сайта и его «фавикон», создать меню и расположить его в необходимой области, разместить разнообразные виджеты, настроить главную страницу как страницу отображения записей или как статическую страницу, а также добавить свои индивидуальные стили для содержимого сайта (*возможна стилизация классов, идентификаторов и тегов HTML*). После проведенных изменений нажмите в окне управления кнопку «Опубликовать», а затем закройте настройки и перейдите на сайт. Проверьте корректность отображения произведенных Вами изменений.

После проведённых манипуляций закройте сообщение в консоли (*для этого существует кнопка «Настройка экрана», при нажатии на которую в опускающемся меню можно снять «галочки» публикации необходимых пунктов*).

1.3 Знакомство с интерфейсом CMS WordPress

Консоль

Основное управление содержимым сайта, построенного с использованием WordPress, осуществляется при помощи меню, расположенного в левой части консоли администратора. Так в пункте меню «Консоль» присутствует два подпункта: «Главная» и «Обновления». Выбрав пункт меню «Главная», Вы всегда сможете перейти на главную страницу консоли администратора, а выбрав

пункт меню «Обновления» попадете в раздел отвечающий за обновление версий самого WordPress, плагинов и тем установленных в данный момент. Так, при построении данного примера, после установки WordPress, в его составе уже шел плагин «Akismet Anti-Spam», но его версия устарела. Перейдя к разделу обновлений можно увидеть, что именно он может быть обновлен до актуальной версии (Рисунок 1.9).

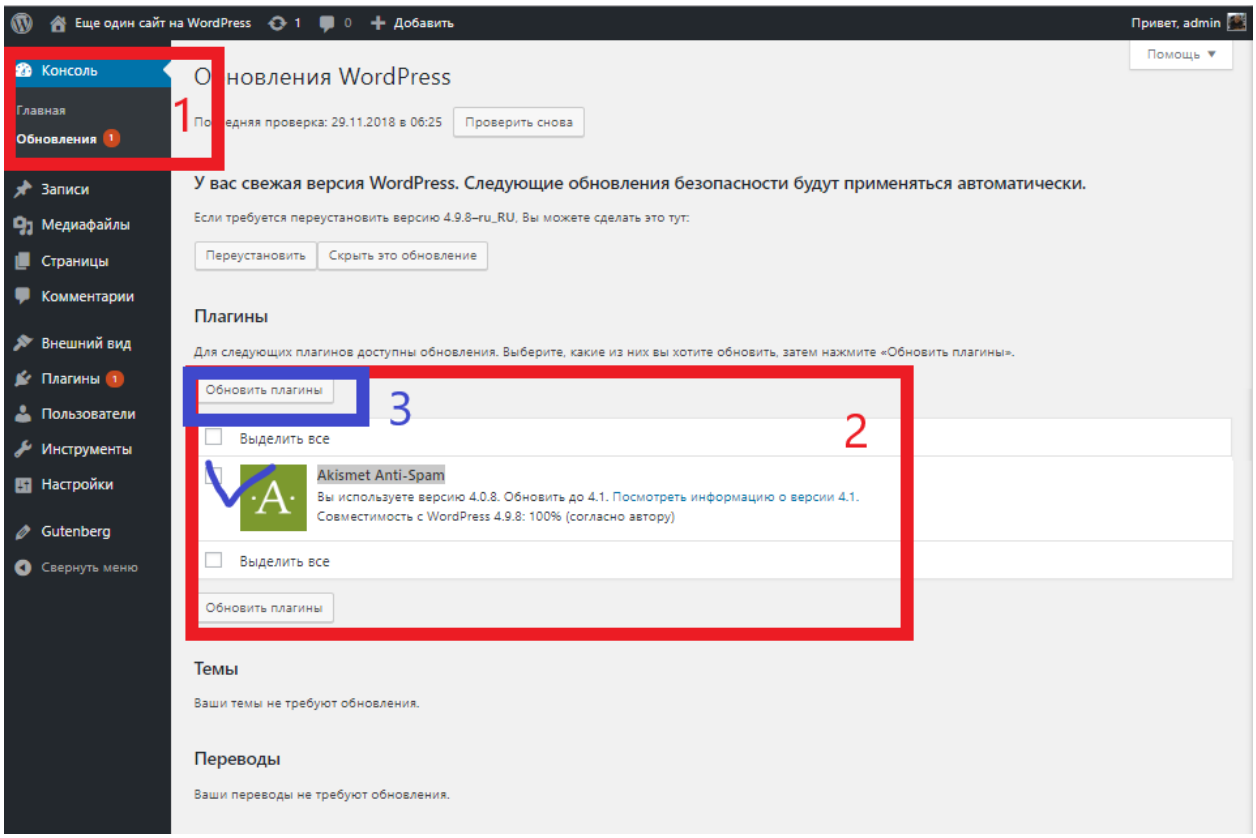


Рисунок 1.9 – Раздел «Обновления» WordPress

Вам необходимо выполнить обновление плагина, для этого напротив его имени необходимо отметить чекбокс и нажать на кнопку «Обновить плагины» (действие 3 на рисунке 1.9). Если бы в списке присутствовали другие плагины требующие обновления, их возможно бы было обновить пакетно, отметив при этом чекбокс «Выделить все».

Записи

В пункте меню «Записи» сосредоточены подпункты, отвечающие за управление записями в виде блога размещенными на сайте. Подпункт «Все записи» позволяет управлять ранее созданными записями (удалять их,

перемещать, изменять, а также получать доступ к записям, удаленным в корзину). здесь же можно прочитать комментарии оставленные к каждой из записей (Рисунок 1.10).

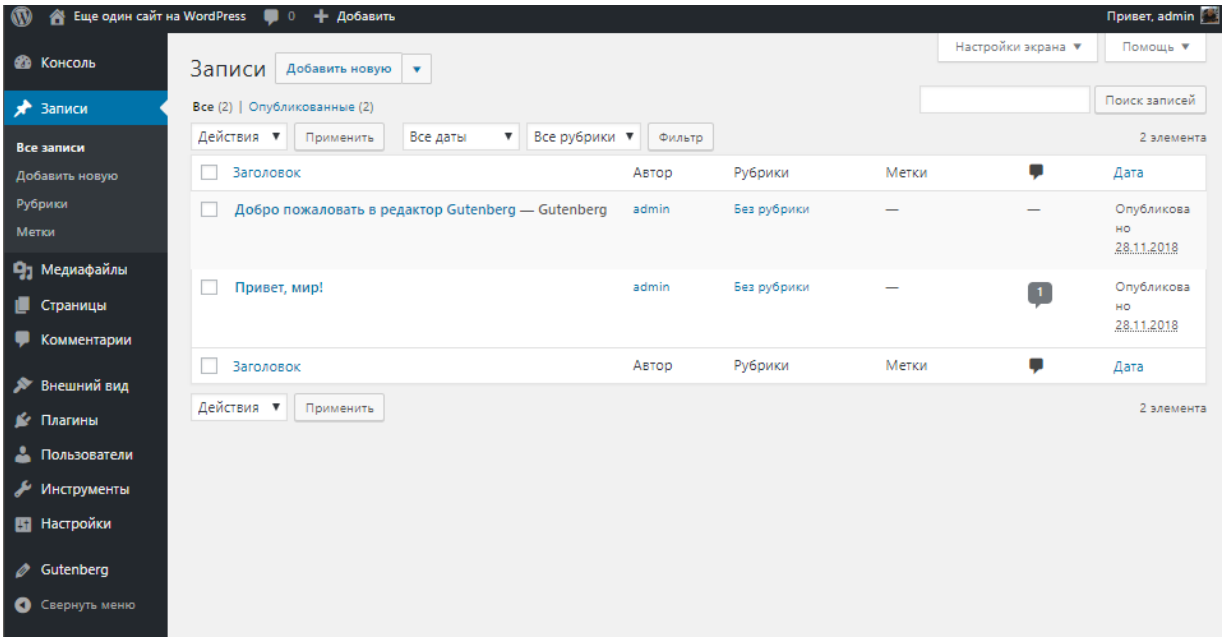


Рисунок 1.10 – Управление существующими записями в CMS WordPress

При помощи подпункта «Добавить новую» создаются новые записи для публикации на сайте. Если выбрать данный подпункт, будет открыта страница добавления записи в которой предоставляются следующие возможности:

- добавление заголовка записи;
- добавление содержимого записи;
- выбор видимости записи (всем, администраторам, только пользователям, имеющим пароль доступа);
- выбор дата и времени публикации;
- выбор рубрики для размещения записи с возможностью добавления новой рубрики из данного окна;
- добавление меток записи (позволяют группировать материалы по наименованиям меток);
- добавление изображения записи (загрузка изображения возможна из того-же окна);

- добавление отрывка записи (вступительный текст или краткое описание записи).

Все перечисленные функциональные возможности перечислены в окне, представленном на рисунке 1.11.

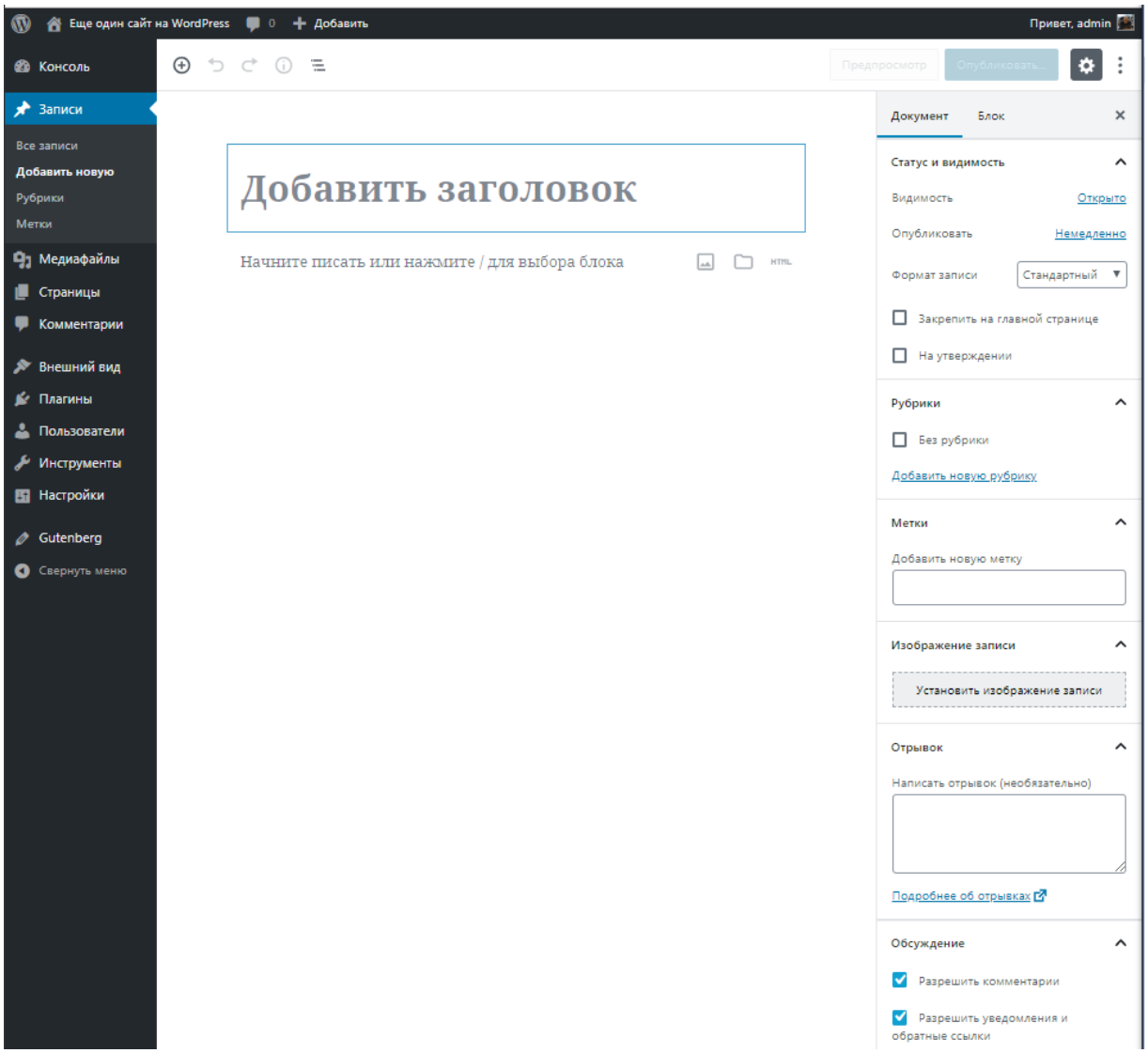
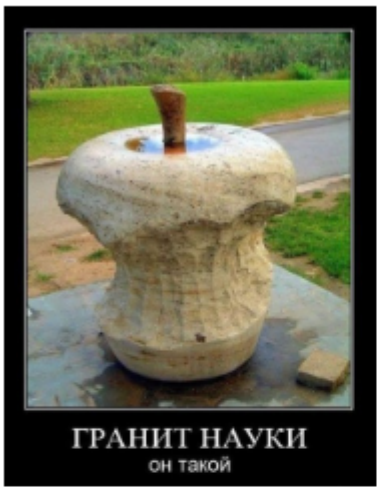


Рисунок 1.11 – Рабочее окно добавления новой записи в CMS WordPress

Создайте свою первую запись в CMS WordPress в соответствии с рисунком 1.12. (*Вы можете использовать случайные изображение и текст, особой важности в этом на данный момент нет*).

29.11.2018 АВТОР: ADMIN

Первая запись сайта на WordPress



Apple через 100 лет

Вот таким образом выглядит запись созданная для CMS WordPress. При написании данной записи использовался редактор кода Gutenberg. В данный блок информации вначале была добавлена картинка и оставлена подпись к ней (Apple через 100 лет). Затем в качестве кода HTML добавлен данный текст. Как видите текст обтекает картинку справа, а сама картинка выровнена по левому краю.

Как только напечатанный текст заканчивается как текст для обтекания картинки, он автоматически начинает занимать всю строку доступную в родительском блоке.

Данный параграф текста как и два параграфа до этого, заключены в теге параграфа «p».

А это заголовок 4 уровня «h4»

```
Зачем мне расстраивать маму,
О двойке своей сообщать?
Из этого сделает драму,
И комп запаролит опять…
А всё из-за цифры… Обидно!
Я только одно не пойму:
Мне - двойка, а Блоку не стыдно
Топить в своей книге Муму?

А это текст оформленный как стихотворение
```

Рисунок 1.12 – Запись добавленная на сайт под управлением WordPress

Медиафайлы

Пункт меню «Медиафайлы» предоставляет доступ к существующей библиотеке медиафайлов и позволяет добавить новые.

В окне библиотеки медиафайлов возможно отображение всех медиафайлов, только изображений, только аудио, только видео, медиафайлы не прикрепленные к записям или страницам, а также вывод по дате добавления.

При помощи кнопки «Добавить новый» происходит добавление новых медиафайлов в CMS WordPress. Максимальный размер загружаемого файла зависит от настроек PHP (по умолчанию в OpenServer это 100 Мб).

Страницы

Пункт меню страницы схож по функционалу с пунктом «Записи». Также возможно управление уже созданными страницами и добавлять новые. В отличии от записей, группировка статей по каким-либо параметрам отсутствует, единственная возможность – сортировка по месяцу создания. На рисунке 1.13 приведен внешний вид окна раздела «Все страницы» консоли администратора WordPress.

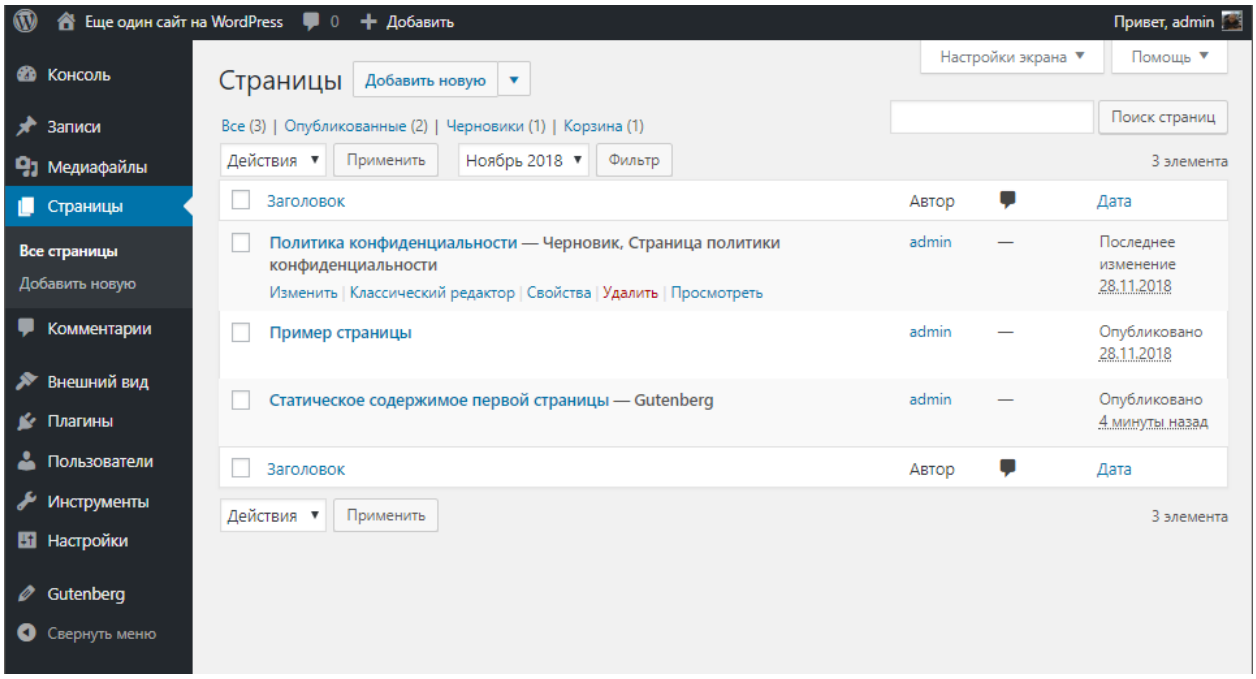


Рисунок 1.13 – Внешний вид окна раздела «Все страницы» консоли администратора WordPress.

Создайте свою первую страницу в CMS WordPress в соответствии с рисунком 1.14.

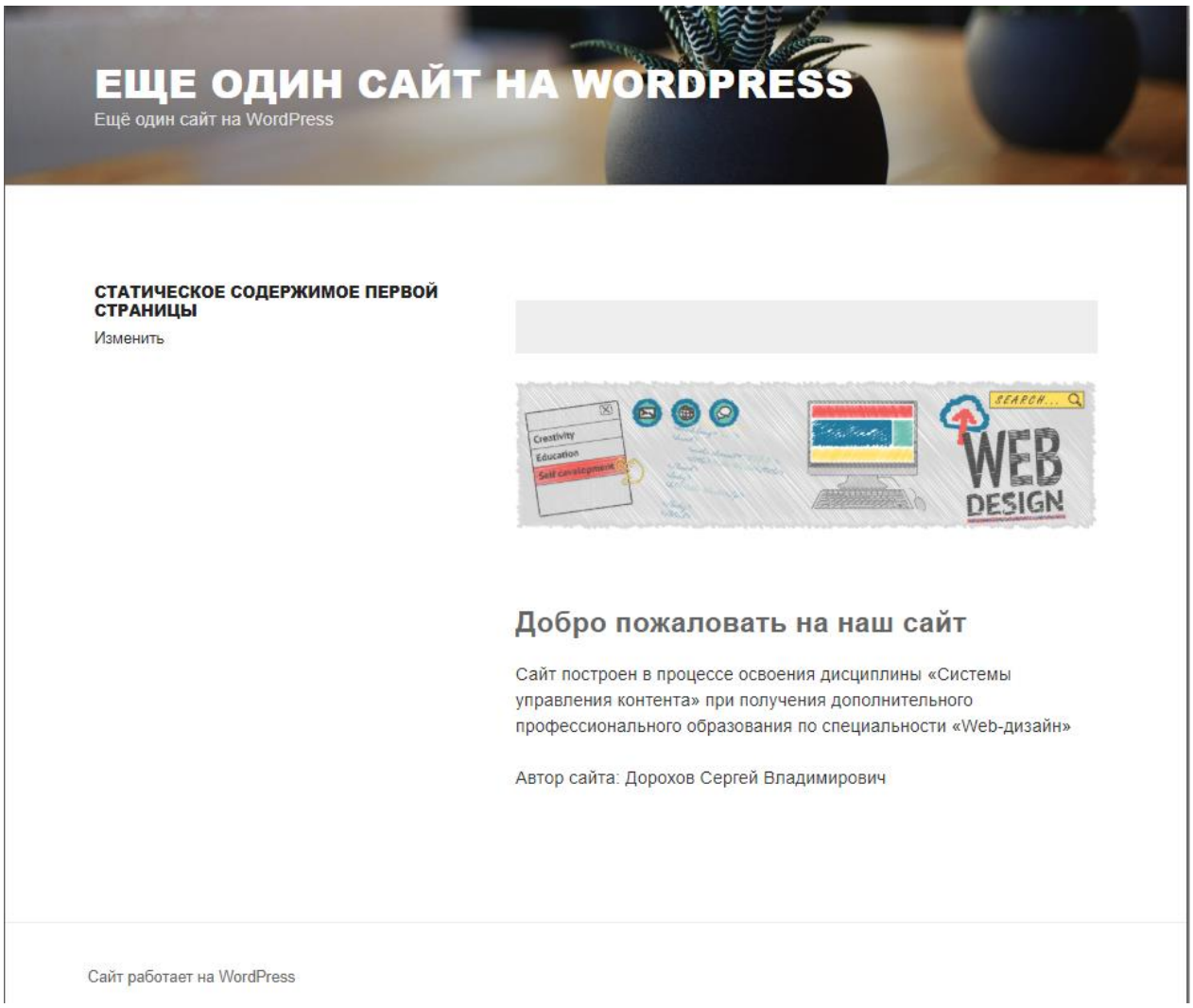


Рисунок 1.14 – Страница добавленная на сайт под управлением WordPress

Комментарии

Раздел «Комментарии» позволяет просмотреть все комментарии, оставленные к опубликованным статьям и записям сайта. На рисунке 1.15 приведен внешний вид раздела «Комментарии» из которого видно функционал раздела. Управление комментариями включает в себя такие действия как одобрение или отклонение комментария, комментарий можно пометить как спам или удалить совсем.

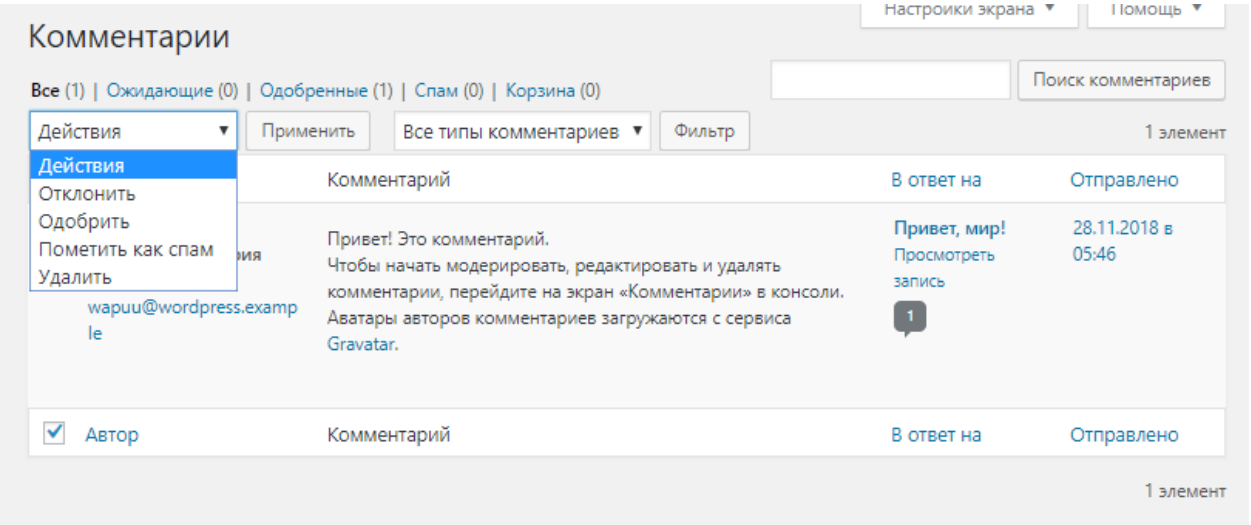


Рисунок 1.15 – Внешний вид раздела «Комментарии» CMS WordPress

Внешний вид

Раздел «Внешний вид» является одним из самых наиболее наполненных по функционалу разделом. Именно из этого раздела производится управление отображением сайта, находящегося под управлением WordPress. Внешний вид любого сайта, в том числе и сайта, построенного с использованием WordPress, очень сильно зависит от используемой темы оформления. Так в WordPress по умолчанию, при установке самой CMS устанавливаются три темы оформления: Twenty Seventeen, Twenty Nineteen и Twenty Sixteen. Помимо этого, в зависимости от используемой темы изменяется и функционал доступный администратору сайта. На рисунке 1.16 приведен внешний вид раздела управления темами сайта, поочередно произведите активацию всех представленных тем, отследите изменения на сайте и в разделе настроить.

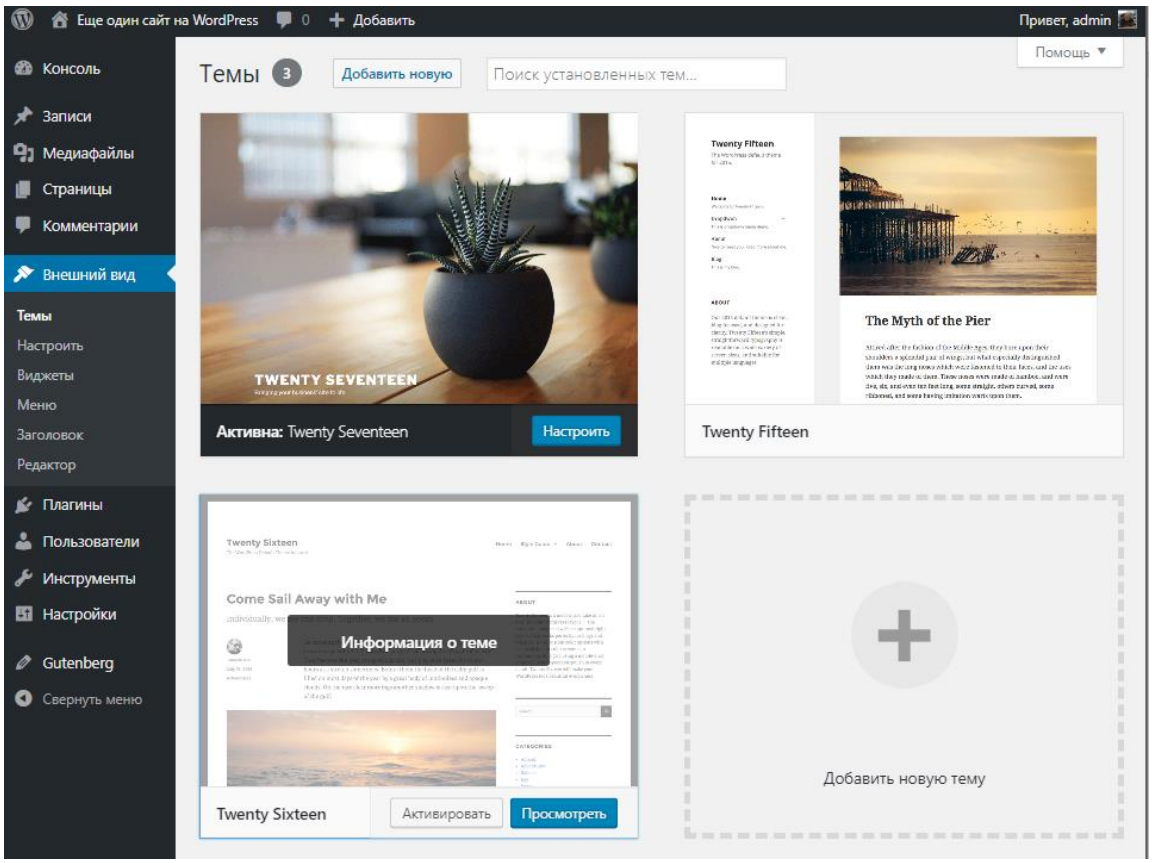


Рисунок 1.16 – Внешний вид раздела управления темами CMS WordPress

При помощи кнопки «Добавить новую» можно добавить в CMS новую тему. Это может быть, как тема уже имеющаяся в базе тем WordPress, так и тема, созданная пользователем. Произведите поиск, установку и активацию темы Ef Practical, имеющейся в репозитории тем WordPress. Отследите изменения, произошедшие на сайте и подразделе «Настроить» раздела «Внешний вид».

Настроить

Функционал подраздела «Настройки» кардинальным образом отличается в зависимости от используемой темы оформления. На рисунке 1.17 приведен внешний вид подраздела для используемой темы оформления Ef Practical. Обратите внимание на тот факт, что непосредственно из данного подраздела консоли сразу можно просмотреть отображение сайта на различных устройствах (функциональные кнопки отмечены на рисунке).

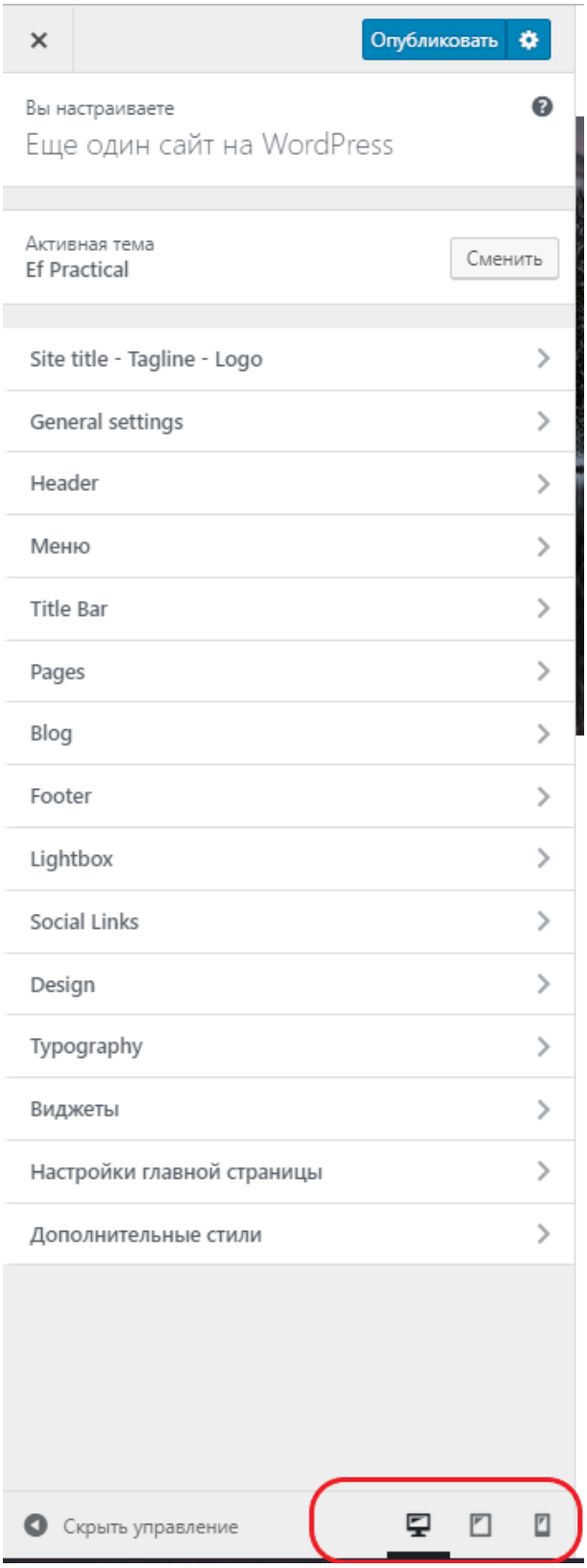


Рисунок 1.17 - Внешний вид подраздела для темы оформления Ef Practical

Виджеты

Подраздел «Виджеты» предназначен для управления отдельными модулями вывода разнообразной информации на все страницы сайта (*например,*

это могут быть виджеты социальных сетей или галерея изображений и видео). На рисунке 1.18, приведен внешний вид окна управления виджетами, как видно их размещение возможно в положения, представленные в правой части окна, а сами виджеты в левой части окна (*из-за огромного количества виджетов и позиций их размещения, данную страницу лучше открыть в консоли администратора, т.к. масштаб рисунка не позволяет оценить весь имеющийся функционал*).

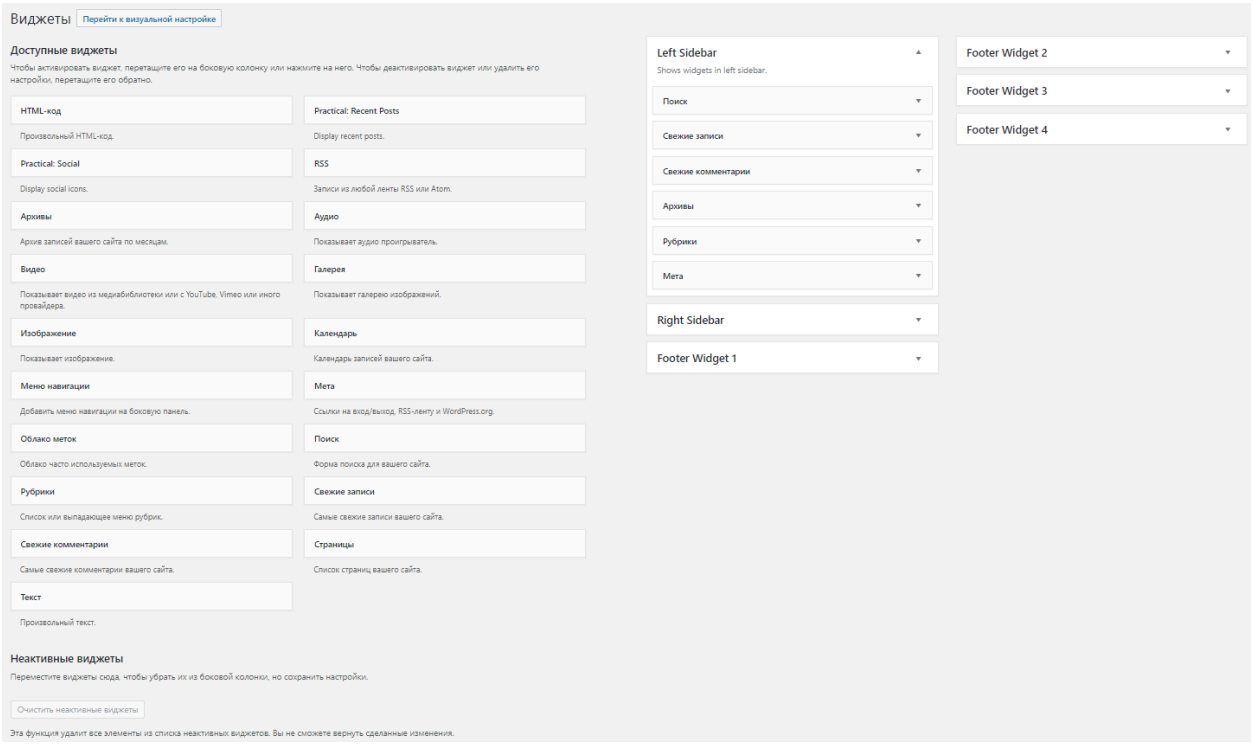


Рисунок 1.18 – Внешний вид подраздела «Виджеты» CMS WordPress

Произведите размещение виджетов «Поиск» и «Календарь» в правом сайдбаре сайта. Произведите их настройку таким образом, чтобы их отображение на сайте соответствовало рисунку 1.19.

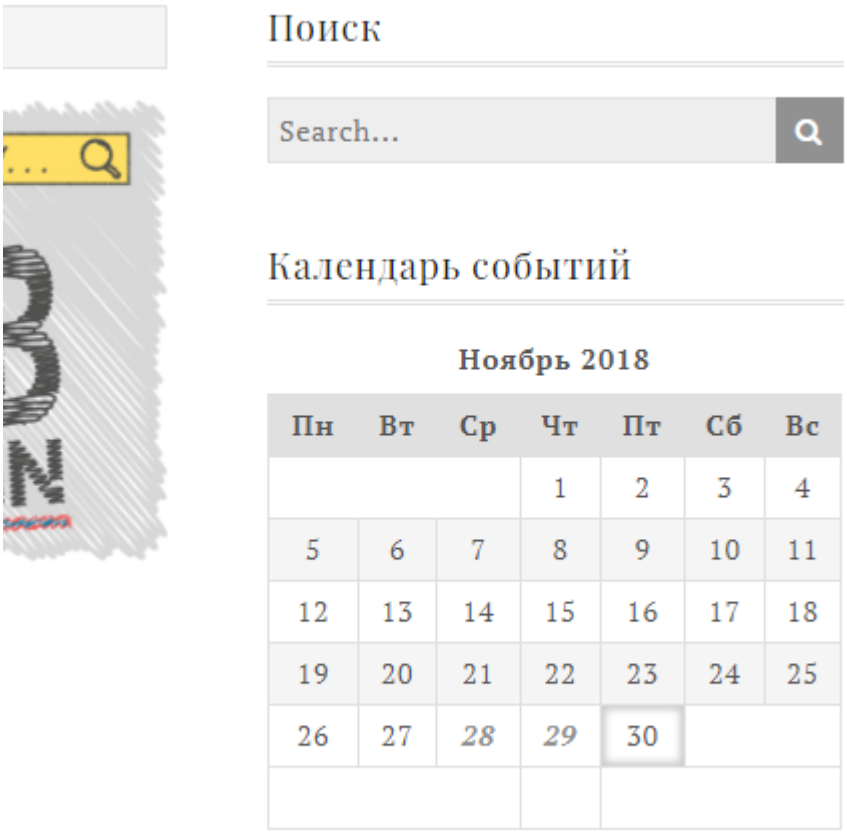


Рисунок 1.19 – Размещённые виджеты «Поиск» и «Календарь» на сайте под управлением CMS WordPress

Меню

Подраздел «Меню» управляет навигацией сайта. Возможно, как управление имеющимся меню, так и создавать новые меню. В качестве пунктов меню возможно использование ссылок на статические страницы WordPress, Записи WordPress, Произвольные ссылки и рубрики записей. На рисунке 1.20 приведен внешний вид окна настроек меню сайта, находящегося под управлением CMS WordPress.

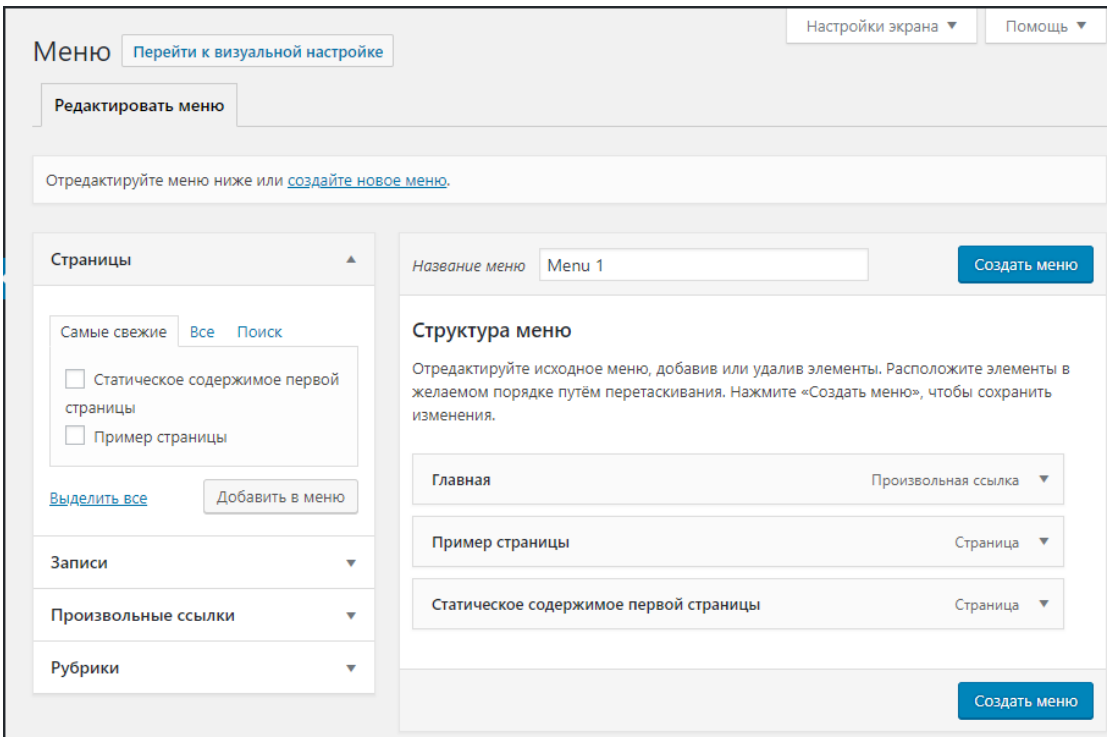


Рисунок 1.20 - Внешний вид окна настроек меню сайта, находящегося под управлением CMS WordPress.

Произведите добавление в меню «Произвольной ссылки» на любой сайт и рубрики «Без рубрики» с наименованием пункта меню «Мои записи»

Заголовок и Фон

Подпункты «Заголовок» и «Фон», ранее уже должны были Вам встретиться в настройках шаблона. Как видно из наименований они отвечают за фон сайта и отображение шапки сайта.

Редактор

Подменю «Редактор» предназначено для редактирования текущей темы оформления сайта. Так как тема состоит из отдельных файлов, редактирование темы заключается в редактировании файлов темы. Так, например, ранее нами был добавлен виджет отвечающий за поиск по сайту, но в его поле по умолчанию в качестве значения атрибута placeholder значиться значение «Search…», а нам его необходимо изменить на значение «Что ищем?». Сделать это можно в файле searchform.php используемой нами темы. Код файла после изменений приведен на рисунке 1.21.

```php
<?php
/**
 * The template for displaying searchform.
 *
 * @package Ef Practical
 */
?>

<form action="<?php echo esc_url( home_url( '/' ) ); ?>" class="search-form" method="get" role="search">
    <input type="search" class="s search-field" name="s" placeholder="<?php esc_attr_e( 'Что ищем?', 'ef-practical' ); ?>">
    <button type="submit" class="search-submit"><i class="fa fa-search"></i></button>
</form><!-- .searchform -->
```

Рисунок 1.21 – Код файла searchform.php темы оформления Ef Practical после изменения

Произведите такое изменение для своего сайта и проверьте корректность отображения на сайте.

Перейдите к пункту меню «Мои записи» сайта, обратите внимание на то, что на странице приведены только части записей и присутствует ссылка «Continue Reading» изменить эту фразу на русскоязычную, например, «Подробнее…» можно в файле entry-content.php, который располагается в папке entry папки template-parts. Произведите самостоятельно такое изменение и проверьте корректность отображения на сайте.

Плагины

Функционал CMS WordPress может быть значительно расширен за счет использования различных плагинов. Также такие плагины как Akismet Anti-Spam позволяют бороться со спамом на страницах Вашего сайта.

В данном разделе консоли происходит управление уже имеющимися плагинами (активация-деактивация, удаление), имеется возможность добавить новый плагин или отредактировать существующий.

Одним из наиболее популярных плагинов форм обратной связи является плагин «Contact Form 7». Произведите его поиск в репозитории, установите и активируйте для своего сайта. После активации плагина в меню консоли появится соответствующий пункт меню для настройки контактных форм.

Пользователи

В данном разделе сосредоточены инструменты управления пользователями их ролями и правами. В CMS WordPress доступны следующие виды ролей пользователей:

- подписчик;
- участник;
- автор;
- редактор;
- администратор;

Ваша учетная запись, которая создана сразу после установки имеет роль «администратор».

Также в данном разделе доступна панель настройки своего профиля. В настройках профиля возможно управление следующим функционалом:

- отключение визуального редактора;
- отключение подсветки синтаксиса при редактировании кода;
- выбор цветовой схемы;
- использование горячих клавиш для проверки комментариев;
- показ верхней панели при просмотре сайта;
- выбор основного языка сайта;
- персональные данные пользователя;
- возможность изменения пароля;
- возможность выхода на других устройствах.

Особого внимания заслуживает использование сервиса Gravatar, для используемого аватара, аватар заданный при регистрации на этом сервисе, будет использоваться везде, где в качестве электронной почты будет указана Ваш адрес (как в консоли администратора, так и в комментариях).

Существенно повышает безопасность сайта на WordPress возможность выхода на других устройствах. Бывает так, что Вам необходимо опубликовать что-то на своём сайте из публичного места, и Вы можете забыть выйти из

консоли администратора на чужом компьютере. Вот в таком случае такая функция будет крайне полезна.

Пройдите регистрацию в сервисе Gravatar, установите свой аватар и проследите его изменение в консоли администратора. Перейдите к любой своей записи и оставьте комментарий, проследите отображение Вашего аватара.

Инструменты

В разделе «Инструменты» собраны функциональные возможности импорта и экспорта данных Вашего сайта в другие системы и из них. Так в данные Вашего сайта могут быть импортированы данные из таких источников:

- Blogger;
- LiveJournal;
- Movable Type и TypePad;
- RSS;
- Tumblr;
- Другая установка Wordpress.

А также имеется встроенная поддержка конвертации рубрик и меток.

Помимо этого, в CMS WordPress имеется возможность экспорта и удаления персональных данных.

Настройки

Раздел «Настройки» содержит в себе следующий функционал:

Общие настройки WordPress

В общих настройках возможно изменение названия и описания сайта, его адреса и адреса почты администратора, имеется возможность включения регистрации пользователей на сайте, определение часового пояса сайта и формата представления времени на сайте.

Настройки публикации

Выбор основной рубрики и формата записей, а также настройки публикации посредством отправки сообщений электронной почты с указанием рубрики публикации.

Чтение

Выбор отображения главной страницы сайта (страница записей или статическая страница). Количество записей публикуемых на страницах записей. Настройки лент RSS. Настройка видимости системы для поисковых систем.

Обсуждение

Настройки комментирования на Вашем сайте. Настройки персонализированы как для статей, так и для записей. Также возможна премодерация комментариев на сайте.

Медиафайлы

Настройки создаваемых размеров копий изображений (миниатюк) для использования на сайте.

Постоянные ссылки

Настройки ссылок сайта. Вид ссылок, создаваемых для новых рубрик, записей и страниц.

Конфиденциальность

Настройки страницы политики конфиденциальности.

Используя ресурсы сети интернет подберите макет верстки используемый в дальнейшем для самостоятельной работы. При осуществлении поиска обратите особое внимание соответствия макета следующим критериям:

- Макет не должен быть одостраничником (LandindPage)$
- Макет должен иметь как минимум два меню (основное и дополнительное) и меню кнопок социальных сетей;
- Макет должен содержать боковую колонку (сайдбар);
- Макет должен содержать примеры страницы и записи с комментариями и формой комментирования;
- В макете должна присутствовать страница обратной связи, содержащая форму для ввода данных.

Контрольные вопросы:

1. Для чего при выполнении данных практических работ используется программа OpenServer?
2. Если CMS WordPress установлена для доменного имени mywp, консоль администратора будет доступна по адресу?
3. Чем, с функциональной точки зрения, отличаются записи и страницы WordPress?
4. В чем преимущество использования в CMS WordPress, тем оформления?
5. При помощи p=какого функционала WordPress расширяются его возможности?

Практическая работа № 2

Создание темы WordPress

Цели работы:

- Научиться создавать минимальную структуру темы для CMS WordPress;
- Научиться подключать файлы к проекту с использованием абсолютных и относительных путей;
- Получить представление о понятиях: событие, экшн и хук используемых в WordPress;
- Научиться регистрировать файлы стилей и скриптов для темы через файл funcions.php.

Оборудование и материалы:

- Персональный компьютер с установленной операционной системой семейства Windows;
- Локальный web-сервер OpenServer;
- Редактор кода SubLime Text 3;
- Сверстанный макет сайта;
- Графический редактор Adobe Photoshop.

Указания к выполнению

2.1 Понимание того, что такое тема

CMS WordPress как и любая другая организована таким образом, что все данные сайта хранится в базе данных (*данные о пользователях, записи, комментарии и прочее*). При посещении сайта построенного на основе CMS WordPress пользователь видит все эти данные через призму темы сайта, т.е. в зависимости от выбранной темы, пользователь увидит данные в том или ином оформлении. Для примера, перейдите в подраздел «Темы» раздела «Внешний вид» консоли администратора и измените тему установленную в данный момент как «активная», а затем обновите сам сайт. Общий внешний вид, как и отображение отдельных элементов сайта должно измениться, в тоже время

содержимое базы данных сайта осталось неизменным. Таким образом, тема является визуальным представлением содержимого сайта. Причем, в зависимости от настроек, примененных к теме, часть данных может присутствовать или отсутствовать в зависимости от выбранной темы.

2.2 Откуда взять дополнительные темы?

Темы WordPress доступны для установки как из магазина тем WordPress, их возможно скачать со сторонних источников и устанавливать в WordPress. Устанавливаемые темы могут быть как платными, так и бесплатными. Как особый вариант стоит рассмотреть создание собственной темы. Все установленные в WordPress темы расположены в подпапке «themes» папки «wp-content» сайта (*соответственно каждая из тем в своей собственной отдельной папке*). Так, на рисунке 2.1 приведено содержимое вышеуказанной папки после свежей установки WordPress. Как видно из данного содержимого, по умолчанию в WordPress установлено три темы:

- twentynineteen;
- twentyseventeen;
- twentysixteen.

Эти же темы Вы увидите, если посмотрите содержимое подраздела «Темы» раздела «Внешний вид» консоли администратора (Рисунок 2.2).

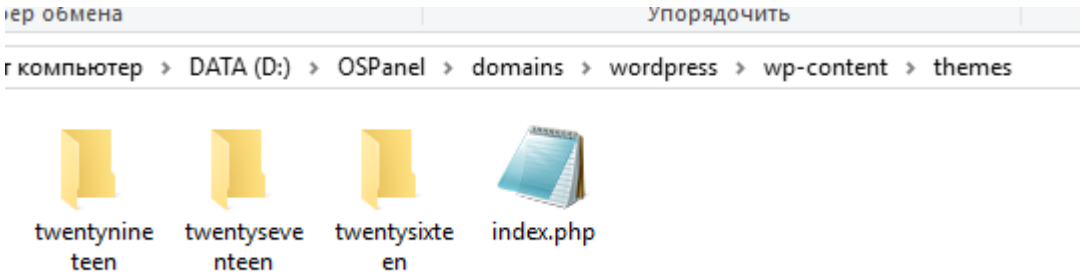


Рисунок 2.1 – Содержимое папки themes после первоначальной установки WordPress

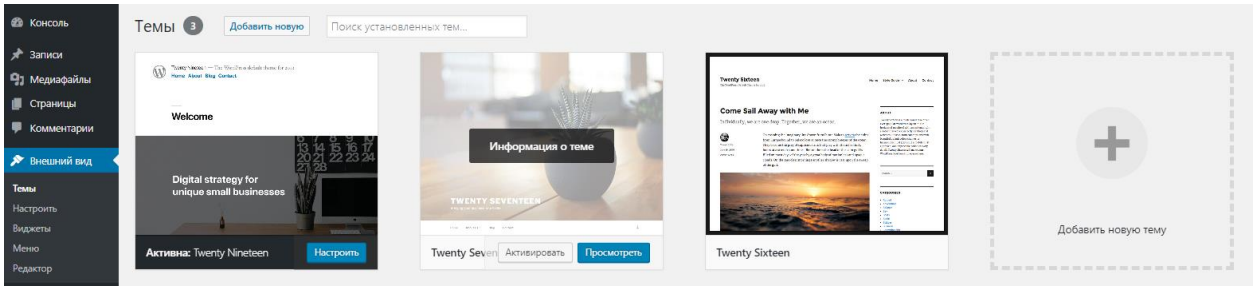


Рисунок 2.2 – Состав тем CMS WordPress после первоначальной установки

2.3 Создание собственной темы

Так как макет рассматриваемой в проекте верстки носит имя «Keep It Simple», это же имя мы будем использовать для именования создаваемого шаблона.

Перейдите в папку «themes» Вашей установки WordPress и создайте папку с именем «keepitsimple», после чего обновите содержимое подраздела «Темы» раздела «Внешний вид». Перед Вами должно появиться сообщение о ошибке в работе темы (рисунок 2.3). В нашем случае это нормально, а сама ошибка говорит о том, что в теме отсутствует файл таблицы стилей.

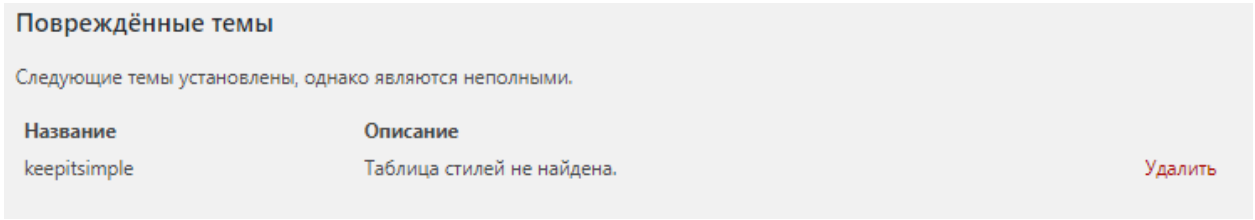


Рисунок 2.3 – Описание ошибки работы темы сайта

Для дальнейшей работы над проектом, нам потребуется открыть папку проекта в любом редакторе кода, например, SublimeText. (*В уже открытой программе выбрать пункт меню File-Open Folder, в открывшемся окне проводника выбрать расположение папки проекта*).

Для исправления ошибки работы темы, в корне проекта (папке «keepitsimple»), необходимо создать новый файл (сочетание клавиш Ctrl+N) и сохранить его с именем «style.css». После этого в очередной раз обновите содержимое подраздела «Темы» раздела «Внешний вид». Ошибка, имеющаяся в

работе создаваемой темы измениться и примет вид, изображенный на рисунке 2.4.

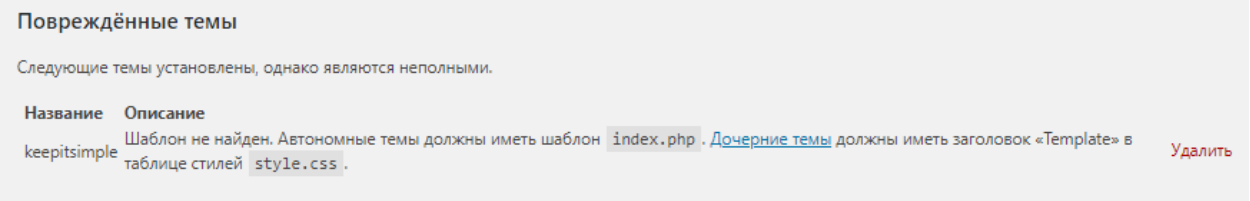


Рисунок 2.4 – Описание ошибки работы темы сайта после добавления файла style.css

Всё дело в том, что помимо созданного нами файла «style.css», в теме должен также присутствовать файл «index.php». Создайте его по аналогии с файлом «style.css», а затем обновите содержимое подраздела «Темы» раздела «Внешний вид». Вы должны увидеть новую тему в перечне отображаемых тем раздела (рисунок 2.5).

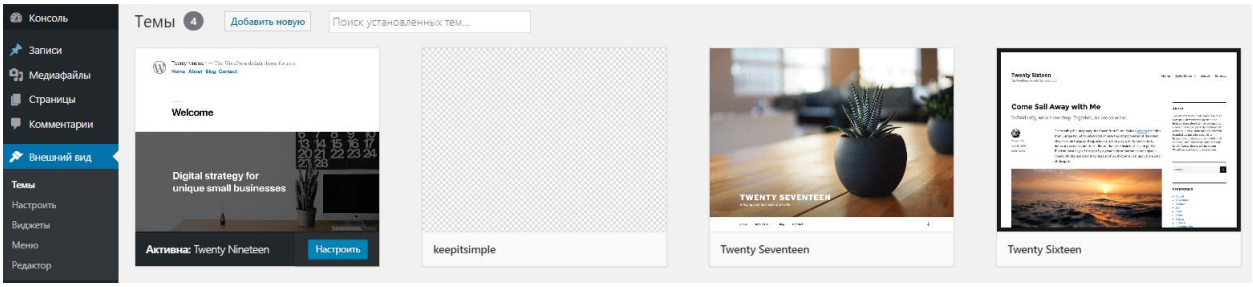


Рисунок 2.5 – Перечень тем после проведённых действий по созданию темы

Активировать и использовать данную тему рано, к тому же она даже не имеет на данный момент описания и отображается она без скриншота. Для устранения этих недостатков нам необходимо:

1. Внести описание темы в файл «style.css»;
2. Сделать скриншот макета;
3. Обрезать скриншот до необходимых размеров (1200x900 пикселей);
4. Сохранить скриншот в папку проекта с именем «screenshot.png».

В файл «style.css» внесите следующее содержимое приведенное на рисунке 2.6.

```
style.css   index.php

1  /*
2  Theme Name: Keep It Simple
3  Author: студент ВГППК
4  Description:  Некоторое описание темы (можете изменить на своё)
5  Requires at least: WordPress 5.0
6  Version: 1
7  */
```

Рисунок 2.6 – Содержимое файла style.css, необходимое для описания темы

Внесенное содержимое будет отображаться только в консоли администратора и никаким образом не повлияет на работу стилевого содержимого, так как оно закомментировано.

Для того, чтобы получить скриншот макета, откройте файл «index.html» предоставленного Вам макета в браузере и нажмите на клавиатуре кнопку «Print Screen» (*может отличаться в зависимости от используемой клавиатуры*). Сделанный скриншот будет находиться в буфере обмена операционной системы.

На следующем шаге, Вам необходимо открыть графический редактор (*например, Adobe Photoshop*), и создать в нем новый документ с размерами 1200 на 900 пикселей. Вставьте в созданный документ имеющийся в буфере обмена снимок экрана и при помощи инструментов графического редактора приведите его в соответствии с рисунком 2.7.

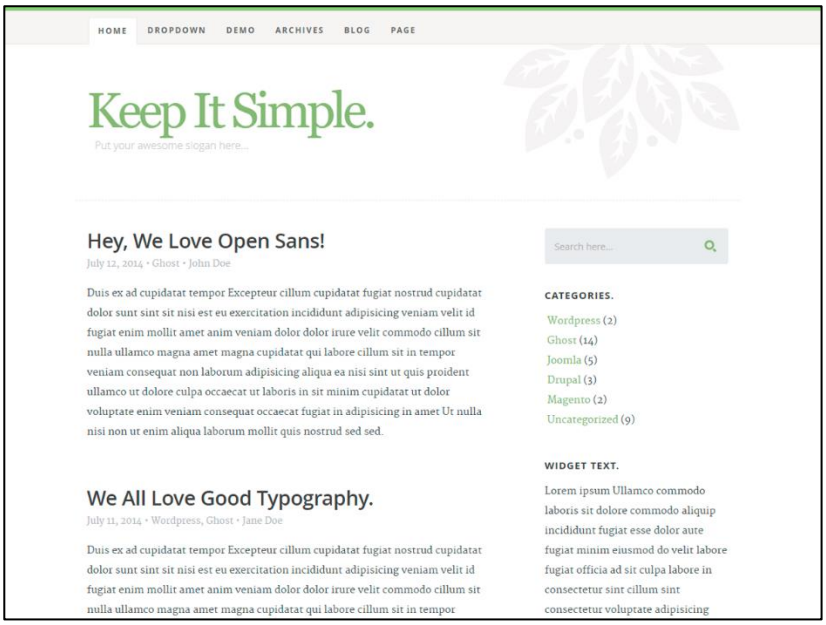


Рисунок 2.7 – Необходимая часть снимка экрана для скриншота темы WordPress

После проведенных работ по обрезке и позиционировании снимка экрана, сохраните файл в папку проекта с именем «screenshot» и расширением «png». Очередной раз обновите содержимое подраздела «Темы» раздела «Внешний вид». вы должны увидеть скриншот темы, а нажав на неё, в отдельном всплывающем окне увидеть ранее созданное описание шаблона.

Активируйте созданную тему и обновите сайт. Так как кроме описания темы и её скриншота в создаваемой теме, ничего нет Вы должны увидеть в браузере «Пустое содержимое». Для того, чтобы на сайте отобразилось хоть что-то нам необходимо это что-то добавить в файл «index.php», например, добавьте в содержимое этого файла свою фамилию и обновите страницу сайта. В моем случае, в браузере отобразилась информация, представленная на рисунке 2.8, в Вашем же случае Вы должны увидеть отображение Вашей фамилии.

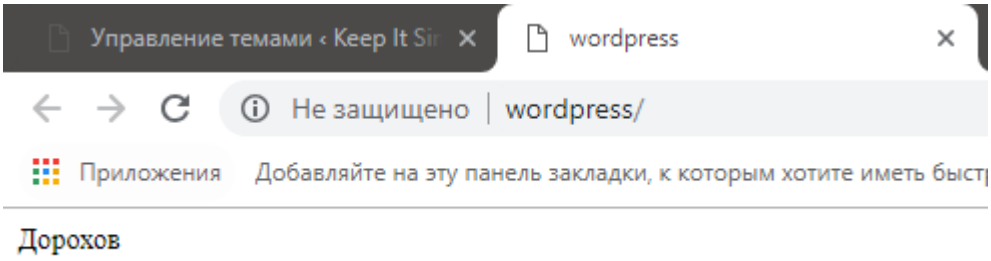


Рисунок 2.8 – Отображение сайта после внесения данных в файл «index.php»

Кроме того, если мы просмотрим сайт в режиме отладчика, то увидим автоматически сформированную структуру html-документа (открывающий и закрывающий тэги html, секции head и body) и сам текст (фамилию).

За формирование стандартных для WordPress шапки сайта и подвала отвечают функции «get_header» и «get_footer» в которых в свою очередь и происходит подключение внешних стилей и скриптов.

Создайте в корне проекта файл header.php, в который добавьте содержимое файла «index.html» макета - от начала файла и до закрывающего тега «header» *(«вырежьте» содержимое из файла*). Создайте в корне проекта файл footer.php, в который добавьте содержимое файла «index.html» макета – от открывающего тега footer и до конца документа (*«вырежьте» содержимое из файла*). В файле

«index.php» вызов двух функций WordPress «get_header» и «get_footer», так как это показано на рисунке 2.9.

Рисунок 2.9 – Вызов функций «get_header» и «get_footer» в файле «index.php»

Обратите внимание на то, что каждая часть кода языка php заключается следующей конструкцией: <?php *часть кода на языке php* ?>, это уже относится к синтаксису языка php.

Между вызовами функций «get_header» и «get_footer» в файле «index.php» вставьте оставшееся содержимое файла index.html.

После обновления страницы сайта на нем должно отобразиться все содержимое, но это содержимое не будет иметь стилевого оформления. Так получилось по нескольким причинам:

- отсутствуют внешние файлы, подключаемые в проекте;
- подключение файлов стилей и скриптов в WordPress, отличается о подключения таких файлов в html.

Все подключаемые к проекту темы WordPres файлы, должны располагаться в папке «assets», создайте такую папку в корне проекта и перенесите в неё папки css, images и js верстки предоставленной Вам для реализации проекта. Но даже после переноса файлов стилей и скриптов, отображение страницы не измениться, так как пути подключения файлов не соответствуют их расположению. Чтобы пути подключения файлов соответствовали их расположению, мы должны их изменить.

Давайте рассмотрим это на примере подключения файла default.css (в файле header.php 21 строка кода). На данный момент подключение файла

происходит из папки css проекта, но в нашем случае расположение папки не соответствует. Мы должны изменить путь к файлу default.css следующим образом:

http://wordpress/wp-content/themes/keepitsimple/assets/css/default.css

Таким образом мы указали абсолютный путь к файлу стилей и после обновления страницы, он должен «подгрузиться» к проекту. Но, такой подход не является правильным по причине того, что мы разрабатываем универсальную для всех сайтов на WordPress тему сайта, и указание абсолютных путей недопустимо, так как при смене адреса сайта все подключенные к нему файлы «отвалятся» и функционал сайта будет нарушен.

В WordPress, для относительного указания расположения файлов темы существует отдельная функция: get_template_directory_uri. Для эксперимента давайте **временно** добавим код представленный на рисунке 2.10 в файл header.php сразу после открытия тега body.

```
<body>
    <?php echo get_template_directory_uri(); ?>
    <!-- Header
```

Рисунок 2.10 – Пример вывода пути к корню текущей темы WordPress при помощи функции get_template_directory_uri

После сохранения файла и обновления странички сайта в браузере, в самом начале страницы выведется путь к корню текущей темы WordPress. Стоит также обратить внимание на то, что оператор «echo», является оператором языка «php», суть его работы заключается в печати одной или нескольких строк текста (более подробно можно почитать в справке по PHP в сети интернет).

Таким образом, изменив путь подключения файла default.css на <?php echo get_template_directory_uri(); ?>/assets/css/default.css мы подключим этот файл с использованием относительных путей расположения файла, при помощи стандартной функции WordPress.

Но к сожалению, в темах WordPress, так не делается. Все подключения файлов и описание функциональных возможностей шаблона WordPress

происходит в отдельном файле темы с именем «functions.php». Создайте такой файл в корне проекта и откройте его для редактирования, а подключение файлов «default.css», «layout.css» и «media-queries.css» из файла «header.php» удалите совсем.

В данном файле мы должны добавить новый «экшн» привязанный к стандартному событию WordPress «wp_enqueue_scripts» и добавить во время его выполнения выполнение собственной функции, которую также описываем в файле «functions.php» (Рисунок 2.11).

Рисунок 2.11 – Добавление «экшена» и выполнение собственной функции при выполнении стандартного события WordPress

Помимо этого, созданное нами подключение к выполнению стандартного события WordPress должно знать после чего ему необходимо выполниться. В нашем случае событие «wp_enqueue_scripts» должно выполниться в момент срабатывания хука «wp_head», и выполнение этого хука мы должны добавить в файл «header.php» перед закрытие тега «head» (Рисунок 2.12).

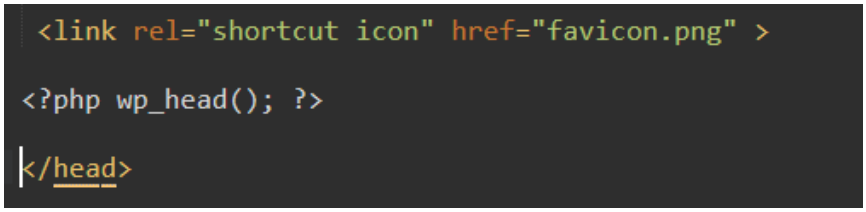


Рисунок 2.12 – Выполнение хука «wp_head» в файле «header.php»

В итоге, после правки файлов «header.php» и «functions.php», файл «default.css» должен корректно подключиться к разрабатываемой теме. Подключение файлов «layout.css» и «media-queries.css» аналогично, а вот подключение главного файла стилей темы незначительно отличается. Так как в корне каждой темы WordPress имеется файл style.css, для указания пути к этому

файлу существует отдельная функция – get_stylesheet_uri. Итоговый код подключения всех файлов стилей к проекту темы приведен на рисунке 2.13.

Рисунок 2.13 – Итоговый код подключения внешних файлов стилей к проекту темы для CMS WordPress в файле «functions.php»

Подобным образом подключаются и внешние файлы скриптов для разрабатываемой темы. Отличия заключаются в том, что они будут выполняться в момент выполнения хука «wp_footer», который мы должны разместить в файле «footer.php» перед закрытием тега «body» (Рисунок 2.14).

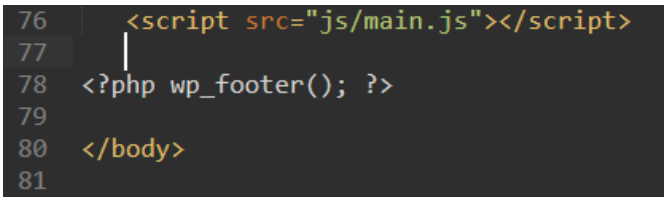


Рисунок 2.14 – Вызов хука «wp_footer» в файле «footer.php»

Также помимо вызова события, мы должны добавить новый экшн, выполняющийся при выполнении хука «wp_footer», с выполнением нашей собственной функции, которая в свою очередь подключит необходимые нам файлы скриптов. В итоге файл «functios.php» примет вид, изображенный на рисунке 2.15.

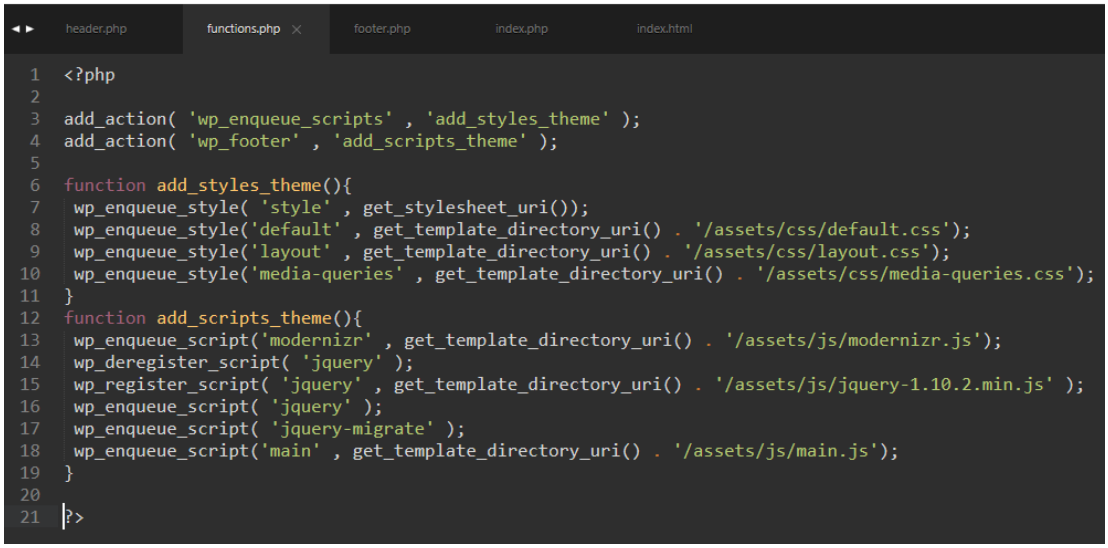


Рисунок 2.15 – Итоговый файл «functions.php» с функциями подключения необходимых файлов стилей и скриптов выполняющихся при событиях «wp_head» и «wp_footer»

Особое внимание необходимо уделить факту отключения встроенной в WordPress версии jquery и подключения версии, идущей вместе со сверстанным макетом. На рисунке 2.15 это произведено в 14-16 строках кода. А вот файл библиотеки jquery-migrate наоборот подключен из состава библиотек, включенных в состав WordPress (*строка 17 кода, изображённого на рисунке 2.15*).

Также обратите внимание на то, что файл «modernizr.js» подключен первым в списке скриптов. В свою очередь в верстке он был подключен в секции «head». Это подключение было сохранено переносе в файл «header.php» и от него необходимо избавиться, удалив соответствующие строки кода в файле «header.php».

В заключении работы посмотрите код сайта во встроенном отладчике браузера и проверьте отсутствие ошибок подключения файлов стилей и скриптов.

Произведите установку CMS WordPress для выполнения самостоятельных работ и для этой установки произведите создание темы на основе макета выбранного для реализации проекта. При проведении работ произведите разбивку индексного файла макета верстки на составные

части (файлы index.php, header.php, footer.php). При разбиении помните, что чаще всего содержимое файлов header.php и footer.php будет повторяться на каждой странице сайта, а содержимое файла index.php будет уникальным для главной страницы будущего сайта.

Произведите правильное подключение файлов стилей и скриптов к проекту (в файле functions.php) не забыв в конце работ проверить функциональную работоспособность сайта и отсутствие ошибок в консоли браузера.

Контрольные вопросы:

1. Что входит в минимальную структуру темы оформления CMS WordPress?
2. Каковы варианты подключения файлов стилей и скриптов к разрабатываемой теме и какой вариант является правильным?
3. Каковы параметры документа, отображающего скриншот главной страницы сайта в консоли администратора?
4. Какие виды хуков существуют в CMS WordPress?

Практическая работа № 3

Автоматизация вывода информации о сайте

Цели работы:

- Научиться выводить информацию о сайте в секции HEAD темы WordPress;
- Научиться выводить информацию о сайте в «Шапке сайта» при разработке темы для WordPress;
- Научиться получать путь к корневой папке сайта на WordPress используя стандартный функционал WordPress.

Оборудование и материалы:

- Персональный компьютер с установленной операционной системой семейства Windows;
- Локальный web-сервер OpenServer;
- Редактор кода SubLime Text 3;
- Сверстанный макет сайта.

Указания к выполнению

В разрабатываемой в проекте теме все META-теги имеют статическое содержимое, кроме того название и описание сайта также выведено на страницы сайта статически. Для решения задач автоматизации вывода требуемой информации из консоли администратора в WordPress существует специальная функция bloginfo()[1]. Так при помощи использования только этой одной функции модно получить на сайте следующую информацию о сайте:

- Название сайта;
- Описание сайта;
- Используемая кодировка;
- Тип контента страницы;
- Текущий язык сайта;
- и прочее.

[1] Более подробно о функции bloginfo можно узнать по адресу: https://wp-kama.ru/function/bloginfo

3.1 Информация о сайте в секции «Head»

Откройте в браузере главную страницу сайта и посмотрите его код в встроенном отладчике браузера (Рисунок 3.1).

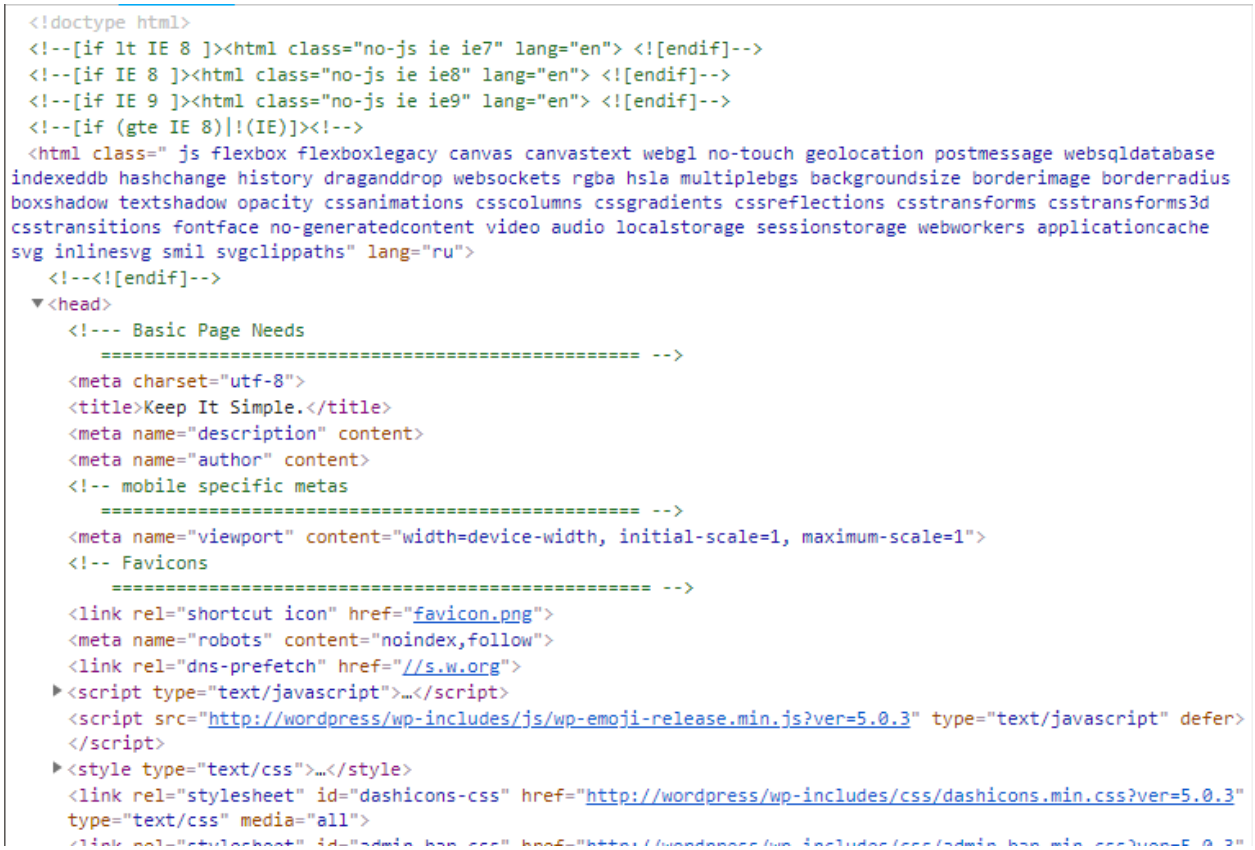


Рисунок 3.1 – Код главной страницы сайта в встроенном отладчике браузера Google Chrome

В развернутой секции head теги заголовка сайта и META-теги, указывающие на кодировку сайта, описание и автора сайта отображаются так, как они были прописаны автором верстки. Нашей задачей является организация вывода этой информации в соответствии с настройками нашего сайта из консоли администратора с использованием функции WordPress – bloginfo().

Откройте в редакторе кода SublimeText для редактирования файл «header.php» и перейдите к первому META-тегу, указывающему кодировку сайта (<meta charset="utf-8">) и модифицируйте его следующим образом: <meta charset="<?php bloginfo('charset'); ?>">. Произведите обновление информации в браузере и проверьте корректность отображения требуемой информации. На

первый взгляд, никаких кардинальных изменений нет, но теперь информация о кодировке сайта берется из настроек WordPress.

В открывающем теге HTML нашего сайта имеется атрибут «lang» со значением «en», эта информация также не корректна и выведена статическим путём. Замените значение атрибута «lang» на следующее: <?php bloginfo('language'); ?> (обратите внимание на то, что тег «html» прописан в комментариях и для устаревших браузеров Internet Explorer, не забудьте изменить это значение и для них).

Значение заголовка сайта в теге «title» замените на: <?php bloginfo('name'); ?>.

Значение атрибута «content» для META-тега со значением атрибута «name» «description» заполните следующим образом: <?php bloginfo('description'); ?>, а META-тег со значением атрибута «name» «author» нам не нужен, удалите строку добавляющую его.

На этом правка вывода информации о сайте в секции завершена

3.2 Информация о сайте теге в «Header» секции «Body»

В том же файле «header.php» в качестве текста заголовка первого уровня с идентификатором «logo-text» статически прописано название макета верстки, замените его на следующий код: <?php bloginfo('name'); ?> (в случае если Ваш сайт называется правильно, отображение информации не изменится, но как и в случае с META-тегами она будет выведена из настроек WordPress).

Кроме того, значение данного заголовка является еще и ссылкой для перехода к статичному файлу «index.html», для того чтобы при клике на название сайта, пользователь был правильно перенаправлен на главную страницу сайта измените для тега ссылки – «a» значение атрибута «href» с «index.html» на следующий код: <?php echo home_url(); ?>. Таким образом в этом коде работают по сути две функции: функция «php» - «echo», которая печатает значение функции WordPress «home_url()».

Содержимое идущего следующим параграфа с идентификатором «intro» также замените на: <?php bloginfo('description'); ?> (так как описание сайта мы не

изменяли, оно отобразится не так как в вёрстке. По желанию вы можете либо оставить значение по умолчанию, либо поправить значение описания сайта в соответствии с макетом).

Добавьте самостоятельно ссылку на главную страницу в файле «footer.php» как показано на рисунке 3.2 используя материалы данной работы.

Рисунок 3.2 – Ссылка на главную страницу сайта в «подвале» сайта

Произведите действия по автоматизации вывода информации о сайте из консоли администратора CMS WordPress для собственной темы используя функции, описанные в данной практической работе.

Особое внимание уделите соблюдению сохранения авторских прав создателя макета верстки используемого Вами в самостоятельном проекте.

Контрольные вопросы:

Какую информацию о сайте можно вывести, используя функционал CMS WordPress?

При помощи какой функции можно получить адрес домашней страницы сайта?

Какие данные могут быть получены при использовании функции bloginfo?

Практическая работа № 4

Регистрация меню в теме для WordPress

Цели работы:

- Научиться регистрировать меню в теме WordPress;
- Научиться вызывать зарегистрированное меню в WordPress;
- Научиться регистрировать несколько меню в WordPress;
- Научиться публиковать в меню сайта различные типы ссылок.

Оборудование и материалы:

- Персональный компьютер с установленной операционной системой семейства Windows;
- Локальный web-сервер OpenServer;
- Редактор кода SubLime Text 3;
- Сверстанный макет сайта.

Указания к выполнению

В настройках внешнего вида сайта WordPress для стандартных тем WordPress присутствует пункт меню «Меню». В нашем же случае данный пункт отсутствует. Для того, чтобы он появился мы должны зарегистрировать меню в файле «functions.php» разрабатываемой темы. За регистрацию меню в WordPress отвечает функция «register_nav_menu», которая в свою очередь должна быть вызвана во время события «after_setup_theme» (время активации темы).

Таким образом, для регистрации меню мы должны добавить новый экшн в файл «functions.php», который во время активации темы выполнит требуемую нами функцию (Рисунок 4.1).

```
4  add_action( 'wp_footer' , 'add_scripts_theme' );
5  add_action( 'after_setup_theme' , 'register_main_menu' );
6
7
```

Рисунок 4.1 – Добавление экшена на возникновение события «after_setup_theme»

Конечно следующим этапом мы должны описать саму функцию «register_main_menu». В этой функции мы как раз и будем регистрировать наше

меню. Внутри созданной нами функции «register_main_menu» выполнится стандартная функция WordPress «register_nav_menu» со значениями параметров $location и $description - 'main_menu' и 'Верхнее меню сайта' соответственно[2] (Рисунок 4.2)

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

function register_main_menu() {
 register_nav_menu( 'main_menu', 'Верхнее меню сайта' );
}

function add_styles_theme(){
```

Рисунок 4.2 – Код функции регистрации меню

После сохранения файлов и обновления консоли администратора, станет доступна возможность управления «Меню сайта», перейдите в данный раздел и создайте меню как показано на рисунке 4.3, после создания меню не забудьте отметить область отображения меню – «Верхнее меню сайта» и сохранить произведенные изменения.

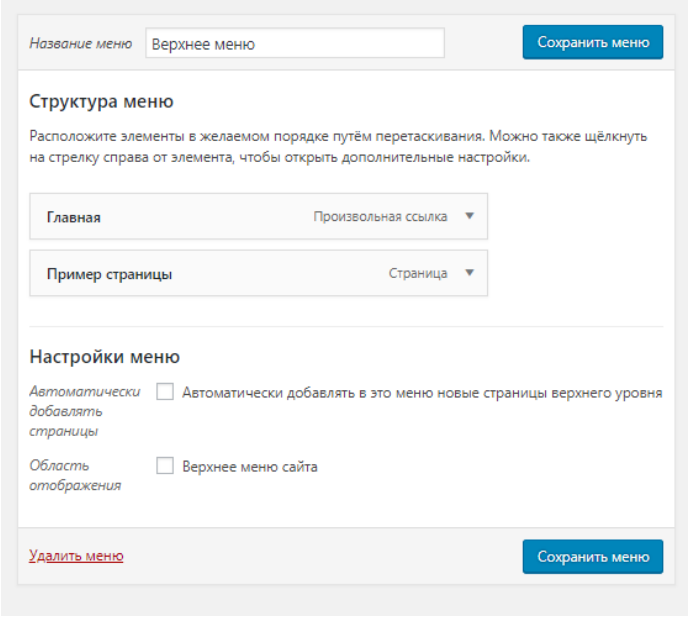


Рисунок 4.3 – Создание меню в консоли администратора

[2] Более подробно можно о работе функции можно узнать по адресу: https://wp-kama.ru/function/register_nav_menu

Как видите область отображения меню перешла из параметров стандартной функции WordPress «register_nav_menu».

На данный момент нам удалось зарегистрировать меню сайта в консоли администратора, но на сайте оно к сожалению, не отображается. Для того, чтобы отобразить меню на сайте мы должны вызвать его в соответствующем месте вывода меню нашей верстки, а точнее в файле где расположена верстка меню – «header.php». Делается это при помощи функции «wp_nav_menu». Добавьте вызов данной функции в файле «header.php» так как показано на рисунке 4.4 и обновите страницу сайта.

```
<a class="mobile-btn" href="#" title="Hide navigation
Hide Menu</a>

<?php wp_nav_menu(); ?>

<div class="row">

  <ul id="nav" class="nav">
```

Рисунок 4.4 – Вызов меню в файле «header.php» разрабатываемой темы

Вызванное меню отобразится на сайте, но без применения стилевого оформления имеющейся верстки. Для исправления данной ситуации нам следует настроить вызов функции «wp_nav_menu»[3]. В нашем случае массив настроек примет вид, изображенный на рисунке 4.5.

```
<?php wp_nav_menu( array(
 'theme_location'  => 'main_menu',
 'container'       => 'div',
 'container_class' => 'row',
 'menu_class'      => 'nav',
 'menu_id'         => 'nav',
 'echo'            => true,
 'fallback_cb'     => 'wp_page_menu',
 'items_wrap'      => '<ul id="%1$s" class="%2$s">%3$s</ul>',
 )); ?>
```

Рисунок 4.5 Настройки функции «wp_nav_menu»

[3] Более подробно можно о работе функции можно узнать по адресу: https://wp-kama.ru/function/wp_nav_menu

Для проверки работоспособности меню 2-го уровня, необходимо создать соответствующую структуру в консоли администратора. Для решения этой задачи отлично должна подойти возможность создавать дочерние и родительские страницы в WordPress. Для начала создайте первую страницу и дайте ей имя «Главная страница подраздела» и опубликуйте её. Контент страницы не имеет особого значения, например, можно добавить текст «Контент главной страницы подраздела». Затем создайте еще две страницы «Первая дочерняя страница подраздела» и «Вторая дочерняя страницы подраздела», соответственно контент этих страниц тоже не имеет значения и его можно по аналогии создать как текст «Контент первой страницы подраздела» и «Контент второй страницы подраздела». При создании дочерних страниц, в редакторе страницы перейдите справа на вкладку «Документ» и в самом низу блока настройки документа укажите в качестве родительской страницу «Главная страница подраздела» (рисунок 4.6).

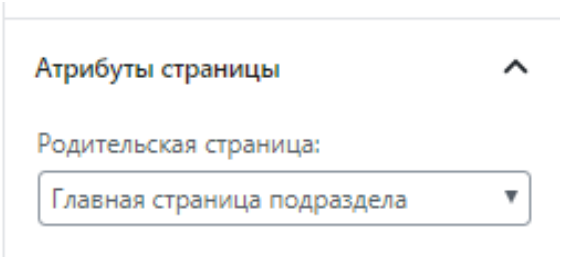


Рисунок 4.6 – Изменение атрибута «Родительская страница»

В итоге содержимое раздела страницы примет вид, изображенный на рисунке 4.7.

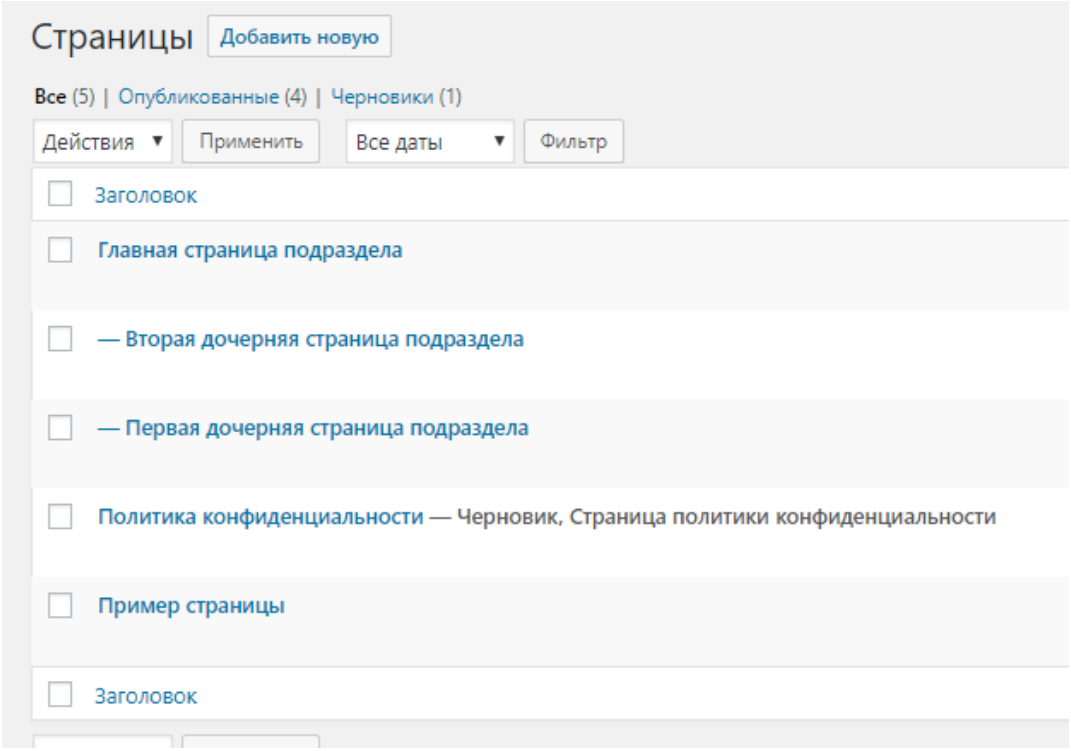


Рисунок 4.7 – Содержимое раздела «Страницы» после добавления требуемых страниц

Для публикации созданных страниц на сайте, необходимо перейти к подразделу «Меню» раздела «Внешний вид». В появившемся меню выделить требуемые страницы и нажать кнопку «Добавить в меню». После добавления страниц в меню дочерние страницы необходимо немного «сдвинуть» вправо, что будет означать дочернюю вложенность страниц, и сохранить изменения в меню. (Рисунок 4.8)

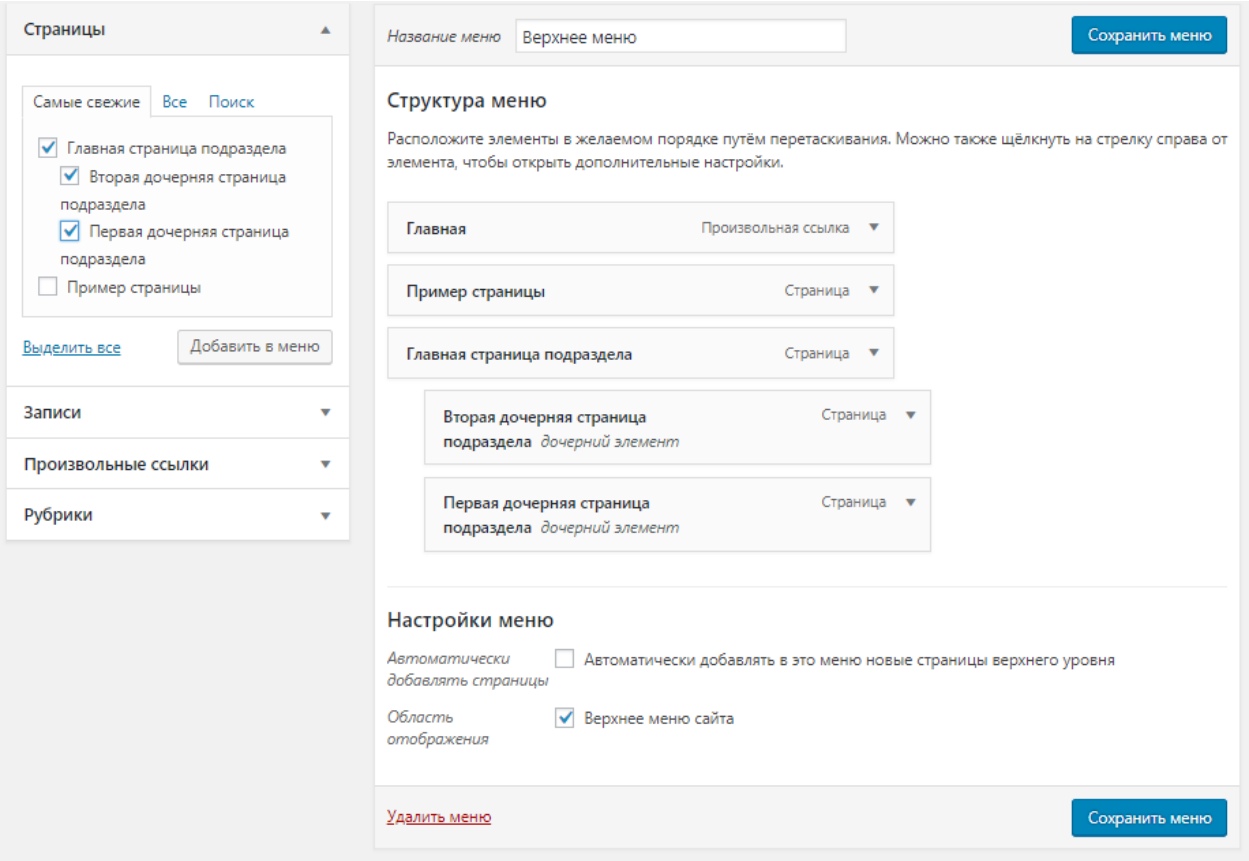


Рисунок 4.8 – Добавление необходимых страниц в меню сайта

После проведенных действий необходимо проверить корректность отображения меню на сайте. В нашем случае, на сайте должно отобразиться меню изображенное на рисунке 4.9, в случае «раскрытого» пункта меню имеющего вложения.

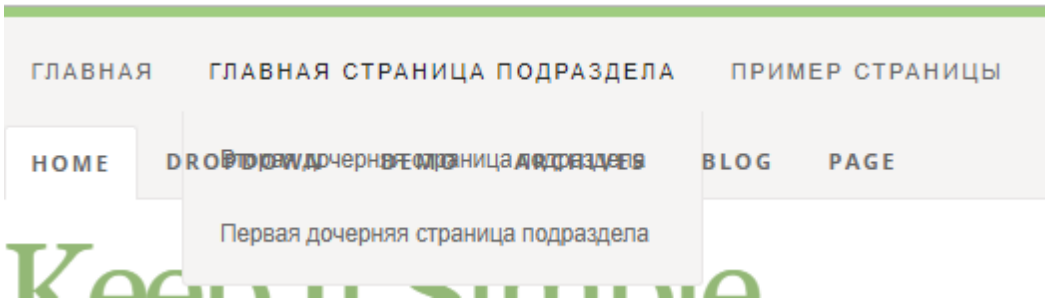


Рисунок 4.9 – Внешний вид меню после регистрации в файле «functions.php» создаваемой темы и добавления в меню требуемых пунктов меню

После проведенных действий и проверки корректности отображения меню на сайте, необходимо удалить статическое меню на сайте (в файле «header.php»

все содержимое блока с классом «row» идущее сразу после описания функции «wp_nav_menu»).

Отличия, связанные начертанием шрифтов, связано с отсутствием поддержки кириллицы в шрифте используемом в стилизации меню сайта, возможно в дальнейшем ситуация может повториться в других элементах, использующих данный шрифт.

Помимо этого, нам необходимо выделить активный пункт меню, дело в том, что у активного пункта меню верстки используется класс «current», а класс сгенерированный WordPress автоматически «current-menu-item». И для того чтобы стилевое оформление применилось к активному пункту меню, в файле «layouts.css» необходимо найти использование класса «current» и заменить его на класс «current-menu-item». Также изменим стили этого элемента, и итоговая стилизация активного пункта меню будет соответствовать рисунку 4.10.

```
167 ul#nav li.current-menu-item > a {
168     font-weight: bold;
169 }
170
```

Рисунок 4.10 – Стилизация активного пункта меню в файле «layouts.css»

После обновления стилей не забудьте сбросить кэш браузера и обновить содержимое страницы (Например, в случае использования Google Chrome в Windows это можно сделать при помощи комбинации клавиш «Shift + F5»).

После стилизации активного пункта меню указанным способом внешний вид меню с выделенным активным пунктом меню примет вид, изображенный на рисунке 4.11.

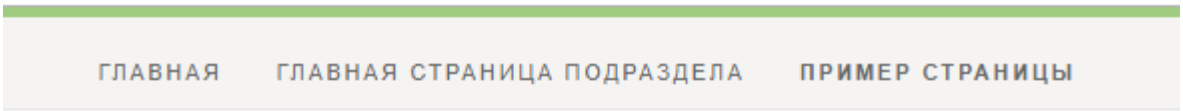


Рисунок 4.11 – Внешний вид меню с выделенным активным пунктом

Для сайта на WordPress, как и, наверное, любой другой CMS, возможен вывод не только одного меню, в случае же WordPress возможность такого вывода должна быть заложена в теме. В подвале нашего макета присутствует еще одно меню содержащее ссылки на страницы Home, Blog, Demo, Archives и About (рисунок 4.12).

Рисунок 4.12 – Внешний вид меню в подвале используемого макета

Для того чтобы вывести данное меню, необходимо в первую очередь иметь соответствующие страницы в перечне страниц сайта. Создайте эти страницы, контент страниц на данном этапе значения не имеет.

После добавления страниц необходимо в файле «functions.php» добавить регистрацию меню подвала (рисунок 4.13).

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

function register_main_menu() {
 register_nav_menu( 'main_menu', 'Верхнее меню сайта' );
 register_nav_menu( 'footer_menu', 'Меню в подвале сайта' );
}
```

Рисунок 4.13 – Регистрация меню подвала в файле «functions.php»

После этого в файле «footer.php» вызвать зарегистрированное меню при помощи функции «wp_nav_menu», указать необходимые опции этой функции и удалить статическое меню из верстки подвала сайта (рисунок 4.14).

```
<div class="two columns">
   <h3 class="social">Navigate</h3>
   <?php wp_nav_menu( array(
            'theme_location'  => 'footer_menu',
            'container'       => null,
            'container_class' => 'row',
            'menu_class'      => 'navigate group',
            'echo'            => true,
            'fallback_cb'     => 'wp_page_menu'
            )); ?>
</div>
```

Рисунок 4.14 - Код вызова меню в подвале сайта

После этого в консоли администратора необходимо создать новое меню, выбрав при этом область отображения и добавить в него требуемые страницы.

Помимо указанных, зарегистрированных и уже вызванных меню в нашем макете присутствует меню социальных сетей. Самостоятельно произведите его регистрацию и вызов на сайте учитывая при этом, что это будут «Произвольные ссылки», а в качестве текста ссылки будут использоваться теги «i» с определенными классами.

Произведите регистрацию всех используемых в верстке меню в файле functions.php и вызов их в соответствующих файлах темы (зависит от используемого макета верстки). Помните, что для вывода страниц, записей, рубрик и прочего контента его необходимо постоянно и методично создавать в консоли администратора. При создании контента учитывайте ориентированность макета верстки.

При адаптации макета основанного на использовании фреймворка Bootstrap будут полезны материалы размещенные по ссылкам:

http://allwordpress.ru/code/vnedryaem-v-wordpress-bootstrap-menyu.html - для версии 3;

https://github.com/jprieton/wp-bootstrap4-navwalker - для версии 4.

Контрольные вопросы:

1. Какова последовательность действий при выводе меню на сайте при использовании CMS WordPress в случае разработки собственной темы?
2. Каким образом вывести различные пункты меню в разных областях сайта?
3. Какова последовательность действия при формировании дочерних страниц и вложенных подменю?
4. При помощи какой комбинации клавиш произвести обновление страницы браузера со сбросом кэша?

Практическая работа № 5

Регистрация и вывод сайдбара в теме для WordPress

Цели работы:

- Научиться регистрировать сайдбар в теме WordPress;
- Научиться вызывать зарегистрированный сайдбар в WordPress;
- Научиться публиковать в сайдбаре различные типы виджетов;
- Научиться корректировке стилей в соответствии со стандартным применением классов WordPress.

Оборудование и материалы:

- Персональный компьютер с установленной операционной системой семейства Windows;
- Локальный web-сервер OpenServer;
- Редактор кода SubLime Text 3;
- Сверстанный макет сайта.

Указания к выполнению

В правой части, используемой нами верстки, используется «Сайдбар». В самой же верстке это код заключенный между открывающим тегом «div» с идентификатором «sidebar» и его закрывающим тегом.

Для того, чтобы наш «сайдбар» был универсален для всех страниц мы должны перенести указанное содержимое из файла «index.php» в новый файл «sidebar.php». Для этого создайте в папке проекта темы новый файл с именем «sidebar.php» и перенесите в него все содержимое тега «div» имеющего идентификатор «sidebar». Обратите также внимание на то, что помимо идентификатора «sidebar» интересующий нас тег «div» использует еще и классы «four» и «columns». Эти классы создают в макете подобие сетки, следовательно, для того чтобы не нарушить отображение на странице в файле «index.php» необходимо вернуть тег «div» использующий эти классы, но не указывать идентификатор, а в файле «sidebap.php» у соответствующего тега избавиться от использования классов (Рисунок 5.1, Рисунок 5.2).

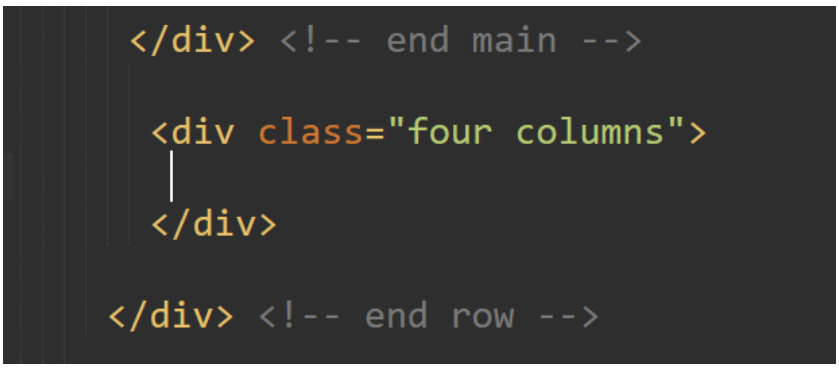


Рисунок 5.1 – Интересующая часть кода файла «index.php» после преобразования

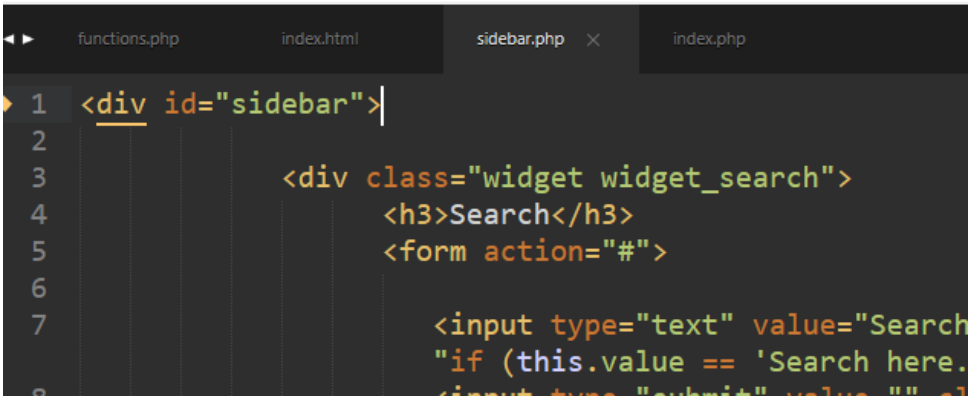


Рисунок 5.2 – Интересующая часть кода файла «sidebar.php» после преобразования

Соответственно в файле «index.php» между открывающим тегом «div» с классами «four» и «columns», которые мы только-что создали, мы должны по аналогии с подключением «шапки» и «подвала» сайта подключить наш сайдбар находящийся в файле «sidebar.php». Содержимое интересующей нас части файла «index.php» примет вид, изображенный на Рисунке 5.3.

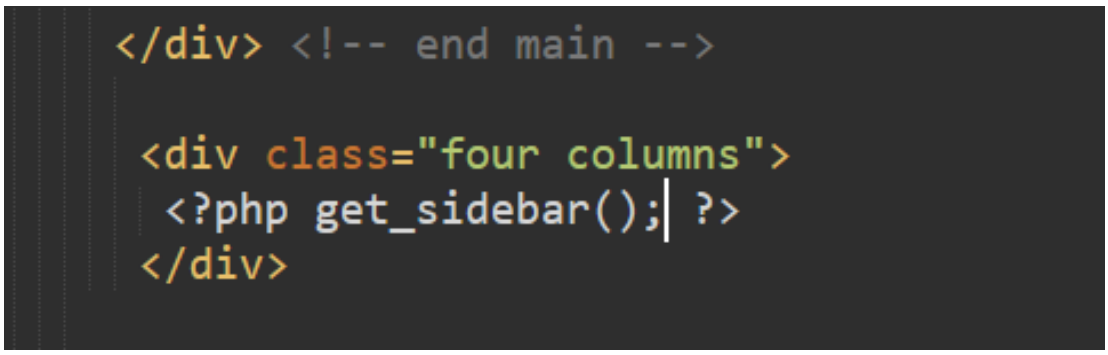


Рисунок 5.3 – Вызов файла «sidebar.php» в файле «index.php»

Если после этого обновить отображение сайта в браузере и сравнить его с первоначальной верстkой, можно заметить некоторые изменения по

отображению элементов, это связано с незначительным несовпадением стилизации и текущей вложенности элементов, а также с отсутствием поддержки кириллицы в используемом вёрсткой шрифте. Позже мы поправим это в файлах стилей.

К сожалению, на данный момент выведенный сайдбар как и его содержимое являются статичными элементами, нам же необходимо вывести эти элементы из консоли администратора в динамическом или автоматическом режиме. Перейдите в консоли администратора к пункту «Внешний вид», на данный момент мы не можем для своей темы управлять виджетами сайта, так как функционал нашей темы этого еще не предусматривает. Так как за функционал нашей темы отвечает файл «functions.php», в нем мы и должны предусмотреть такие возможности. Для этого мы должны добавить новый экшн, который будет выполняться во время выполнения стандартного события WordPress – «widgets_init» и выполнит функцию «register_theme_sidebar» (Рисунок 5.4)

```
add_action( 'widgets_init' , 'register_theme_sidebar');

function register_main_menu() {
```

Рисунок 5.4 – Добавление экшена регистрации сайдбара

Сама же функция примет вид, изображенный на рисунке 5.5.[4]

```
function register_theme_sidebar(){
 register_sidebar( array(
  'name'          => 'Right Sidebar',
  'id'            => "right_sidebar",
  'description'   => 'Сайдбар в правой части сайта',
  'class'         => '',
  'before_widget' => '<div id="%1$s" class="widget %2$s">',
  'after_widget'  => "</div>\n",
  'before_title'  => '<h3 class="widgettitle">',
  'after_title'   => "</h3>\n",
 ) );
}
function register_main_menu() {
```

Рисунок 5.5 – Функция регистрации сайдбара

[4] Более подробно можно о работе функции можно узнать по адресу: https://wp-kama.ru/function/register_sidebar

В данном коде:

«Right Sidebar» - имя сайдбара отображаемое в консоли администратора;

«right_sidebar» - идентификатор, по которому можно обратиться именно к этому сайдбару (а на сайте их может быть и несколько);

«Сайдбар в правой части сайта» - описание сайдбара в консоли администратора;

« » - содержимое начала «обертки» каждого виджета;

« \n» - содержимое окончания «обертки» каждого виджета;

«<h3 class="widgettitle» - содержимое начала заголовка каждого виджета;

«</h3>\n» - содержимое окончания заголовка каждого виджета.

После проведенных действий станет доступной возможность добавлять виджеты в сайдбар разрабатываемой темы., соответствующий пункт в консоли администратора можно будет увидеть после обновления странички консоли.

Перейдите в соответствующий раздел консоли и добавьте в созданный сайдбар виджет «Поиск», дайте ему заголовок «Search». После перехода к сайту и обновления содержимого можно заметить, что ничего в его отображении не изменилось. Это произошло по причине того, что мы не осуществили вызов нашего сайдбара в файле его вывода (файл «sidebar.php»). Для исправления этой ситуации приведите код файла «sidebar.php» к коду, изображенному на рисунке 5.6.

Рисунок 5.6 – Итоговый код файла «sidebar.php» после внесения изменений

В данной конструкции кода мы вызвали динамический сайдбар с идентификатором «right_sidebar», который ранее указали при написании функции.

Теперь после обновления страницы сайта мы можем увидеть, что в сайдбаре выведен виджет поиска по сайту с именем «Search».

Для стилизации данного виджета согласно верстки необходимо изменить содержимое файла стилей «layuot.css». Откройте его и найдите в нём описание стилей, указанное на рисунке 5.7

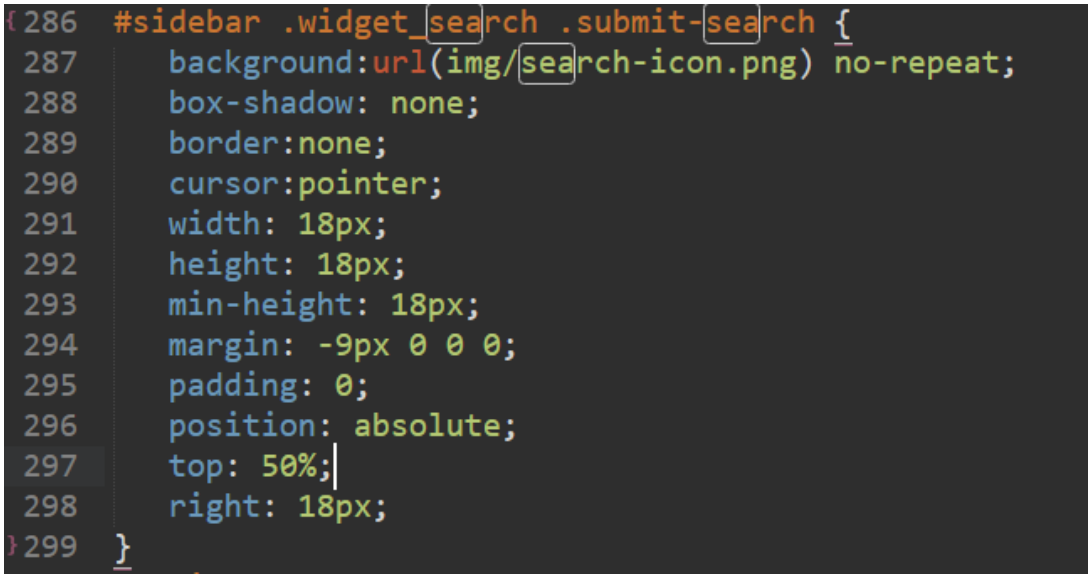


Рисунок 5.7 – Фрагмент стилизации кнопки виджета поиска

Добавьте к имеющимся свойствам, свойства «font-size» и «line-height» с нулевыми значениями, а саму последовательность обращения к элементам замените на «#sidebar .widget_search #searchsubmit».

Ниже данного описания элементов добавьте описания, представленные на рисунке 5.8

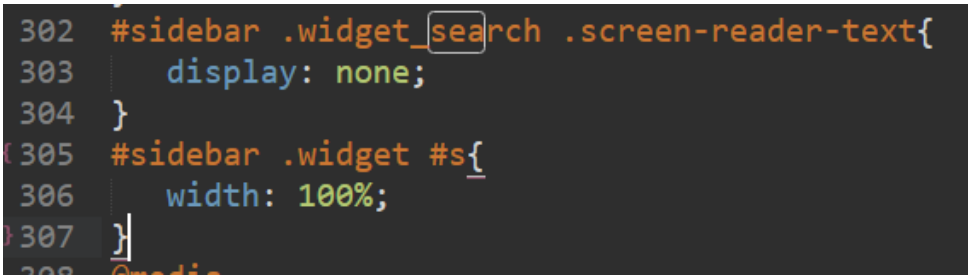


Рисунок 5.8 – Дополнительные элементы стилизации для виджета поиска

Помимо виджета поиска, в WordPress существуют и другие виджеты. Некоторые из виджетов уже стилизованы в используемой вёрстке, а некоторые нет. Самостоятельно произведите добавление:

виджета «Рубрики» с отображением числа записей;

виджета «Облако меток» - с таксономией «Метки»

При необходимости произведите их стилизацию в соответствии с макетом. Помните при этом, что будет необходимым добавление на сайт некоторого количества записей с указанием рубрик и используемых тегов. Для облегчения поиска информации для добавления на сайт вся информация представлена в архиве media-5.zip.

Для приведения стилей в соответствии с макетом, необходимо будет воспользоваться встроенным в браузер отладчиком.

Произведите перенос содержимого сайдбара сайта в отдельный файл (sidebar.php) и его вызов в основном файле темы. Произведите регистрацию динамического сайдбара и его вызов в файле sidebar.php.

Добавьте для вывода на сайте виджеты используемые в верстке и убедитесь в корректности отображения. В случае несоответствия произведите перестилизацию элементов или их новую стилизацию.

Контрольные вопросы:

1. Назначение сайдбара в теме WordPress и на сайте в целом?
2. Каково отличие статического и динамического содержимого темы WordPress?

Практическая работа № 6

Шаблоны контента в теме для WordPress

Цели работы:

- Сформировать представление о назначении отдельных файлов темы, отвечающих за вывод информации;
- Освоить применение стандартного цикла WordPress;
- Научиться выводить пагинацию по страницам для различных рубрик записей;
- Научиться выводить на страницу записи используя функцию WordPress get_posts.

Оборудование и материалы:

- Персональный компьютер с установленной операционной системой семейства Windows;
- Локальный web-сервер OpenServer;
- Редактор кода SubLime Text 3;
- Сверстанный макет сайта.

Указания к выполнению

На данном этапе нами сформирована тема для CMS WordPress, в которой динамически формируются меню сайта и его сайдбар. Содержимое же страниц остается неизменным при переходе к различным пунктам меню. Происходит это потому, что в нашей теме отсутствуют соответствующие файлы, назначение которых в WordPress стандартизировано, а именно:

- index.php – шаблон главной страницы, выводятся короткие анонсы статей;
- page.php – шаблон статических страниц;
- single.php – шаблон постов (статей);
- category.php – шаблон рубрик;
- search.php – шаблон выдачи результатов поиска;
- 404.php – шаблон сообщения о несуществующей странице;

- comments.php – шаблон комментариев.

Так как файл index.php у нас уже существует давайте продолжим его доработку.

В рассматриваемом макете верстки на главной странице сайта предусмотрен вывод последних записей (постов). Для того чтобы WordPress мог выводить эти посты, в файле «index.php» мы должны проверить существование этих постов. Сам же вывод постов по коду верстки начинается в блоке с идентификатором «main» и классами «eight» и «columns», и осуществляется в тегах «article» с классом «entry».

Для вывода же постов сайта динамически, мы должны запустить стандартный для WordPress цикл проверки существования постов, а в качестве примера мы выведем заголовки всех имеющихся в данный момент постов (Рисунок 6.1).

```
6   <div class="row">
7
8    <div id="main" class="eight columns">
9     <?php if(have_posts()) {while (have_posts()){ the_post(); ?>
10     <article class="entry">
11     <header class="entry-header">
12      <h2 class="entry-title">
13       <?php the_title(); ?>
14      </h2>
15     </header>
16     </article>
17    <?php }
18    }?>
19    <article class="entry">
20
```

Рисунок 6.1 – Вывод заголовков записей (постов) на странице «index.php» при помощи стандартного цикла WordPress

После внесения изменений в файл «index.php» и обновления страницы сайта в браузере, на главной странице сайта должны быть выведены заголовки всех записей (постов) имеющихся в WordPress на настоящее время. В данном коде за вывод самих заголовком отвечает функция «the_title();».

За макет вывода записи (поста) мы должны взять содержимое одного из тегов «article» с классом «entry», а уже непосредственно в его верстку включить

функционал WordPress. При помощи функций WordPress мы сможем вывести не только заголовки записей (постов) но и остальное содержимое, например, дату создания записи (поста), её автора, краткое описание и многое другое[5].

Так проведя изменение кода вывода записи (поста) для случая использования нашей верстки, и удаления из неё статического вывода записей (постов) код файла «index.php» примет вид, изображенный на рисунке 6.2

index.php

```
<?php get_header(); ?>
   <div id="content-wrap">
    <div class="row">
     <div id="main" class="eight columns">
      <?php if(have_posts()) {while (have_posts()){ the_post(); ?>
       <article class="entry">
        <header class="entry-header">
         <h2 class="entry-title">
          <a href="<?php the_permalink(); ?>" title="Ссылка на: <?php the_title_attribute(); ?>"">
           <?php the_title(); ?>
          </a>
         </h2>
         <div class="entry-meta">
          <ul>
           <li><?php the_time('j F Y') ?></li>
           <span class="meta-sep">&bull;</span>
           <li><a href="#" title="" rel="category tag"><?php the_category(', '); ?></a></li>
           <span class="meta-sep">&bull;</span>
           <li><?php the_author_posts_link() ?></li>
          </ul>
         </div>
        </header>
        <div class="entry-content">
         <?php  the_excerpt(); ?>
        </div>
       </article>
      <?php } 
      }?>
     </div>
     <div class="four columns">
      <?php get_sidebar(); ?>
     </div>
    </div>
   </div>
<?php get_footer(); ?>
```

Рисунок 6.2 – Код файла «index.php» выводящий на главную страницу сайта все имеющиеся записи (посты)

Данная конструкция является стандартной для WordPress, она выводит на страницу все записи (посты) не обращая внимания ни на какие дополнительные параметры самих записей, а ограничивает только количество вывода. Изменить количество выводимых постов можно в консоли администратора в подразделе

[5] Более подробно о стандартном цикле WordPress и используемых при выводе записи (поста) функциях можно ознакомиться по ссылке: https://wp-kama.ru/id_119/the-loop.html

«Чтение» раздела «Настройки». По умолчанию значение параметра «На страницах блога отображать не более» установлено равное 10, а нашей верткой предусмотрено вывод на главную страницу 3-х последних записей. Измените значение указанного параметра на значение 3, сохраните изменения и обновите страницу сайта. Как можно убедиться при таких настройках сайт наиболее полно соответствует имеющейся в распоряжении верстке. К сожалению, изменение настроек количества вывода записей (постов) на страницах сайта имеет глобальное значение для всего сайта, а нас это не устраивает, так как на других страницах возможно потребуется вывод другого количества записей (постов). Поэтому вывод постов на главной странице сайта при помощи стандартного цикла WordPress нельзя считать самым удачным, а для подобных ситуаций в WordPress существует дополнительный функционал. Сама же получившаяся страница «index.php» нам пригодится, так как она является страницей, отвечающей за вывод записей (постов) из определенной рубрики или отсортированных по определенным тегам. Создайте копию страницы «index.php» и переименуйте её как «category.php».

Очень часто бывает так, что в той или иной рибрике имеется достаточно большое количество записей (постов) и каждый раз изменять параметры для отображения всех записей рубрики на странице неправильно, для решения таких задач существует пагинация страниц, когда все огромное количество страниц разбивается на страницы с определенным количеством записей (постов), а под ними существует навигация по таким страницам.

Для обеспечения такого функционала в разрабатываемой теме, измените окончание стандартного цикла WordPress так, как показано на рисунке 6.3[6]

[6] Более подробно о работе функции можно узнать по ссылке: https://wp-kama.ru/function/the_posts_pagination

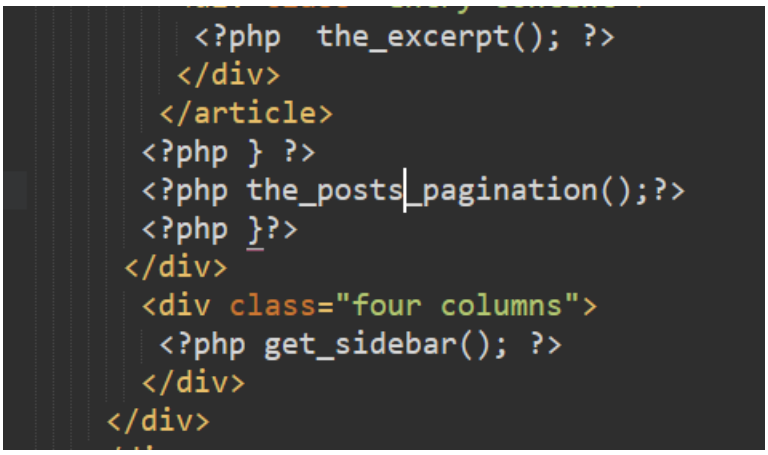


Рисунок 6.3 – Вывод пагинации на страницах разрабатываемой темы

После внесения изменений и обновления страницы с выводом записей (постов) число которых больше числа указанного в настройках чтения, внизу раздела вывода записей (постов) появится навигация по страницам вывода записей (постов) определенной рубрики (рисунок 6.4).

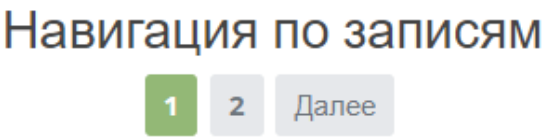


Рисунок 6.4 - Пагинация на страницах разрабатываемой темы

Как видно из рисунка 6.4 над пагинацией присутствует заголовок «Навигация по записям». Если просмотреть код страницы во встроенном в браузер отладчике, этот заголовок сформирован при помощи тега <h2> (тег языка html означающий использование заголовка второго уровня). С точки зрения SEO – оптимизации (да и исходя с точки зрения стилевого представления сайта) этот тег вместе с содержимым лучше скрыть (или убрать). Для решения такой задачи можно воспользоваться «хуком», являющийся фильтром по своему типу, а добавить этот «хук» необходимо в файл функций темы «functions.php» (Рисунок 6.5).[7]

[7] Код для данного примера взят по адресу: https://wp-kama.ru/function/the_posts_pagination. По этому же адресу Вы можете ознакомиться с другими примерами работы над пагинацией по записям сайта

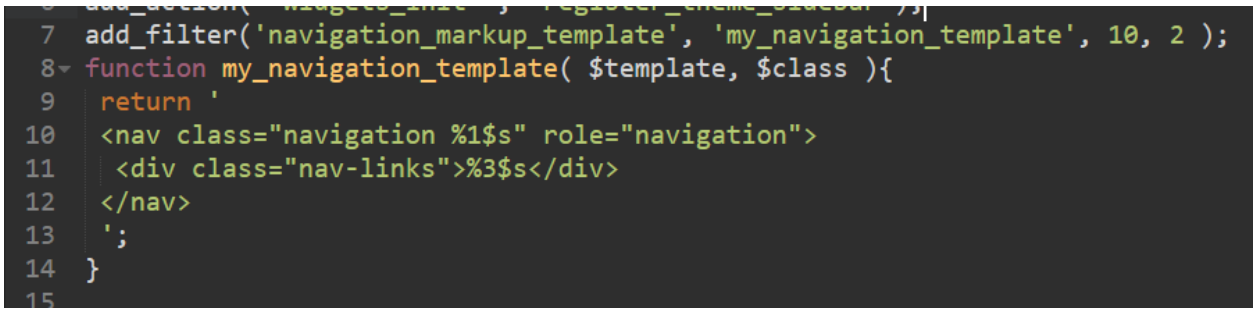


Рисунок 6.5 – Использование «хука» удаляющего заголовок из вывода пагинации записей сайта

На данном этапе правка файла отвечающего за формирование страницы выводящей статьи из определенной рубрики завершена, и нам необходимо вернуться к правке файла «index.php» отвечающего за вывод главной страницы сайта.

Как видно из макета, над которым мы ведем работу на главную страницу сайта выводятся три последних записи добавленных на сайт (точнее их заголовок, информация о дате публикации, авторе публикации, рубрика к которой относится запись и вступительный текст записи), при этом на странице отсутствует пагинация и мы не хотим ограничивать количество записей выводимых на страницах рубрик в консоли администратора.

Для осуществления таких возможностей в файле «index.php», вместо стандартного цикла WordPress нам необходимо воспользоваться возможностями функции WordPress – «get_posts()»[8]. По адресу https://wp-kama.ru/function/get_posts подробно рассказано о работе и опциях данной функции, перейдите по указанной ссылке и скопируйте «Шаблон использования». Вставьте скопированный код в файл «index.php» сразу после блока с идентификатором «main» обернув код тегами кода PHP (Рисунок 6.6).

[8] По адресу: https://wp-kama.ru/function/get_posts можно подробно ознакомиться с работой данной функции

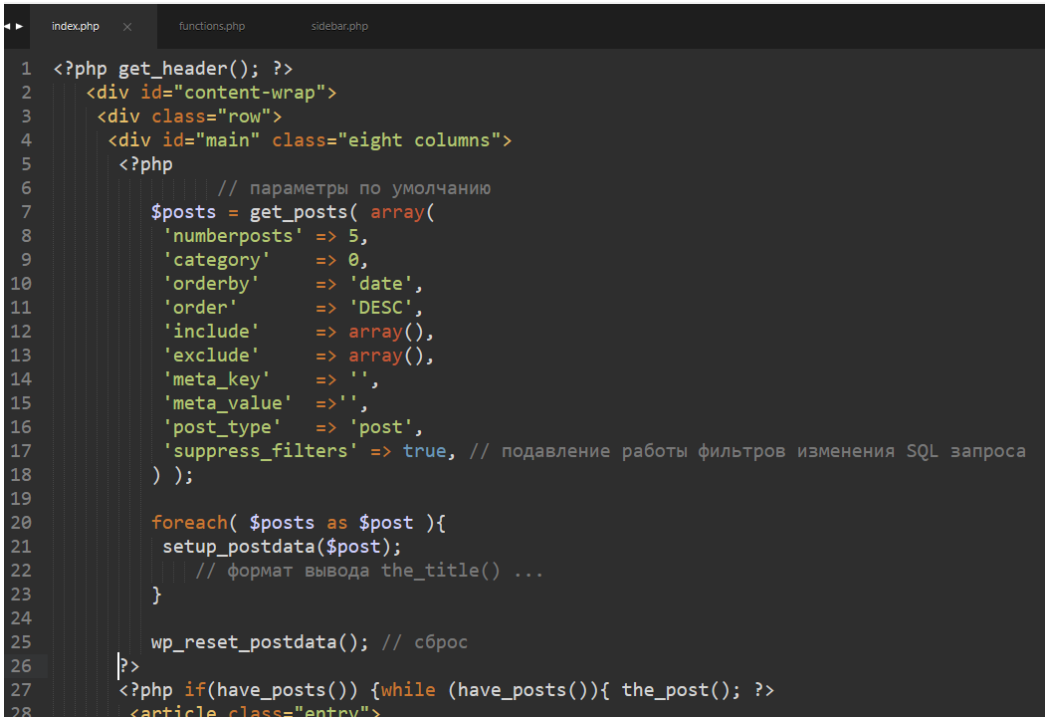


Рисунок 6.6 – Использование функции «get_posts» в файле «index.php» разрабатываемой темы

В данном шаблоне использования идет объявление переменной «posts», в которой перечислены некоторые настройки и их значения, о всех возможных к использованию параметрах и принимаемыми ими значениями можно ознакомиться в справочных материалах, указанных ранее, нам же необходимо изменить только значение параметра «numberposts» со значения 5 на 3. Именно по параметрам, указанным в данном массиве параметров будут отсортированы все записи сайта.

Следующим по коду идет цикл «foreach» в котором и идет вывод постов в соответствии с указанными в ранее описанном массиве параметрами. Как видно из кода, формат вывода записей в шаблоне использования не описан, поэтому мы должны описать его сами. Для этого нам необходимо «разорвать» выполнение кода PHP и перенести всё содержимое ранее подготовленного тега «article» использовавшегося ранее в стандартном цикле WordPress (Рисунок 6.7).

```
    'suppress_filters' => true, // подавление работы фильтров изменения SQL запроса
) );

foreach( $posts as $post ){
 setup_postdata($post);
    ?>
    <article class="entry">
     <header class="entry-header">
      <h2 class="entry-title">
       <a href="<?php the_permalink(); ?>" title="Ссылка на: <?php the_title_attribute(); ?>"">
        <?php the_title(); ?>
       </a>
      </h2>
      <div class="entry-meta">
       <ul>
        <li><?php the_time('j F Y') ?></li>
        <span class="meta-sep">&bull;</span>
        <li><a href="#" title="" rel="category tag"><?php the_category(', '); ?></a></li>
        <span class="meta-sep">&bull;</span>
        <li><?php the_author_posts_link() ?></li>
       </ul>
      </div>
     </header>
     <div class="entry-content">
      <?php  the_excerpt(); ?>
     </div>
    </article>
    <?php
}

wp_reset_postdata(); // сброс
?>
```

Рисунок 6.7 – Изменение формата вывода поста в функции «get_posts»

Код же идущий сразу после используемой функции «get_posts». являющийся кодом стандартного цикла WordPress (На рисунке 6.8 выделен в редакторе кода), нам больше не нужен, удалите его.

```
49      <?php if(have_posts()) {while (have_posts()){ the_post(); ?>
50
51      <?php }
52      }?>
```

Рисунок 6.8 – Код стандартного цикла WordPress, подлежащего удалению из файла «index.php»

После внесения всех изменений в файл «index.php» произведите его сохранение и обновите главную страницу сайта. Проверьте правильность и корректность отображения информации.

Добавьте в состав рубрик записей Вашего сайта новую рубрику «Мониторы» и используя материалы, представленные в архиве «media-6.zip»,

создайте новую запись, относящуюся к данной рубрике. Отследите изменения на главной странице сайта:

- добавленная запись должна отобразиться первой на главной странице сайта;
- в виджете отображающем рубрики сайта должна появиться новая рубрика;
- метки добавленные при создании новой записи должны отобразиться в составе меток выведенных пир помощи соответствующего виджета.

В случае корректности и правильности отображения информации на сайте работу над файлом «index.php» - отвечающим за вывод главной страницы сайта можно считать завершенной

Произведите действия по созданию файла, отвечающего за вывод записей по рубрикам (category.php). Не забудьте реализовать функциональные возможности пагинации страниц вывода. В случае необходимости произведите вывод требуемого количества записей на главную страницу используя возможности функции get_posts (зависит от используемого макета верстки).

Контрольные вопросы:

1. Структура темы WordPress? Назначение отдельных файлов, входящих в состав темы?
2. Что такое «стандартный цикл WordPress»? Каково его отличие от использования функции get_posts?
3. На каких страницах принято выводить пагинацию по страницам? В каком файле темы логичнее всего использовать возможности использования пагинации?

Практическая работа № 7

Шаблоны вывода страниц и записей в теме для WordPress

Цели работы:

- Сформировать представление о назначении отдельных файлов темы, отвечающих за вывод информации;
- Получить практический опыт расширения функционала темы;
- Научиться выводить комментарии к записям и форму для добавления комментариев;
- Научиться осуществлять фильтрацию полей формы добавления комментариев.

Оборудование и материалы:

- Персональный компьютер с установленной операционной системой семейства Windows;
- Локальный web-сервер OpenServer;
- Редактор кода SubLime Text 3;
- Сверстанный макет сайта.

Указания к выполнению

На данном этапе мы научились выводить посты в виде списка как при помощи стандартного цикла WordPress, так и при помощи функции get_posts, но вывод отдельных постов еще не возможен. Как и оговаривалось ранее, за вывод отдельных записей в теме WordPress отвечает файл «single.php», создайте такой файл в корне темы, а в качестве содержимого файла добавьте содержимое представленное на рисунке 7.1.

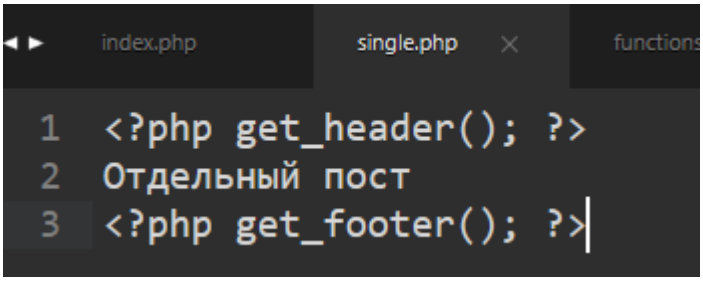


Рисунок 7.1 – Начальное содержимое файла single.php

После создания и сохранения файла, осуществите переход к любой отдельной записи сайта, Вы должны увидеть подключенные header и footer сайта, а также содержимое «Отдельный пост». Как видите WordPress корректно отреагировал на появление нового файла в теме, при попытке открыть отдельный пост (запись) сайта, она вывелась при помощи файла «single.php».

Для корректного отображения оставшихся элементов страницы, мы должны восстановить разметку страницы так как это было задумано разработчиком вёрстки. Для этого откройте в редакторе Sublime Text файл «single.html» используемой вёрстки и скопируйте в нём весь код от открытия блока с идентификатором «content-wrap» до открывающего тега «footer» и вставьте его в файл «single.php» разрабатываемой темы вместо фразы «Отдельный пост» (Полезными будут оставленные автором вёрстки комментарии). Произведите вывод динамически сформированного сайдбара (так как это было сделано ранее для страницы «index.php») и обновите страницу. Перед Вами откроется страница отдельного поста, но его содержимое статически прописано так, как это было в верстке.

Так как содержимое блока отвечающего за вывод заголовка поста и его мета-данных идентично этому же содержимому в файле «index.php» мы можем его скопировать и заменить в файле «single.php» (рисунок 7.2.).

```
<header class="entry-header">
 <h2 class="entry-title">
  <a href="<?php the_permalink(); ?>" title="Ссылка на: <?php the_title_attribute(); ?>"">
   <?php the_title(); ?>
  </a>
 </h2>
 <div class="entry-meta">
  <ul>
   <li><?php the_time('j F Y') ?></li>
   <span class="meta-sep">&bull;</span>
   <li><a href="#" title="" rel="category tag"><?php the_category(', '); ?></a></li>
   <span class="meta-sep">&bull;</span>
   <li><?php the_author_posts_link() ?></li>
  </ul>
 </div>
</header>
<div class="entry-content">
```

Рисунок 7.2 – Код отвечающий за формирование заголовка поста

Теперь, после обновления страницы Вы должны увидеть «Правильный» заголовок поста, дату его публикации, рубрику к которой он относится, но не

увидите автора поста. Все дело в том, что не получен весь массив данных поста, так как мы его не вызвали. Для устранения этого упущения добавьте вызов массива данных поста, сразу после открытия блока для вывода контента на странице «single.php» (см. Рисунок 7.3).

Рисунок 7.3 – Вызов функции the_post на странице поста

После внесенных изменений и обновления страницы сайта, имя автора поста также отобразится. На этом этапе мы закончили вывод заголовка поста со всеми его мета-данными. Следующим блоком, идущим по вёрстке, является блок вывода изображения записи, но данная возможность отсутствует в нашей теме. Для устранения этого недостатка, необходимо добавить поддержку данной функции в файле «functions.php». Нас интересует функция WordPress «add_theme_support»[9]. Данная функция должна сработать в момент активации темы и у нас уже существует функция, выполняющаяся при активации темы, эта функция регистрирует наши меню на сайте. Добавьте в содержимое функции «register_main_menu» (файл «functions.php»), вызов функции «add_theme_support» со значениями «post-thumbnails» и «array('post')» (Переменные «post-thumbnails» и «array('post')» отвечают за вывод изображений записей и эти записи имеют формат постов, т.е. для страниц вывод изображения будет недоступен) (см. Рисунок 7.4).

[9] Подробнее о работе функции можно узнать по ссылке: https://wp-kama.ru/function/add_theme_support

```
}
function register_main_menu() {
 register_nav_menu( 'main_menu', 'Верхнее меню сайта' );
 register_nav_menu( 'footer_menu', 'Меню в подвале сайта' );
 register_nav_menu( 'social_menu', 'Меню социальных сетей' );
 add_theme_support( 'post-thumbnails', array('post') );
}
function add_styles_theme(){
```

Рисунок 7.4 – Вызов функции «add_theme_support» со значением «post-thumbnails» во время регистрации меню сайта

После произведенных действий, необходимо перейти к редактированию любой из записей в консоли администратора WordPress, в секции отвечающей за свойства всего документа, появится новая функциональная возможность – добавления изображения записи (рисунок 7.5).

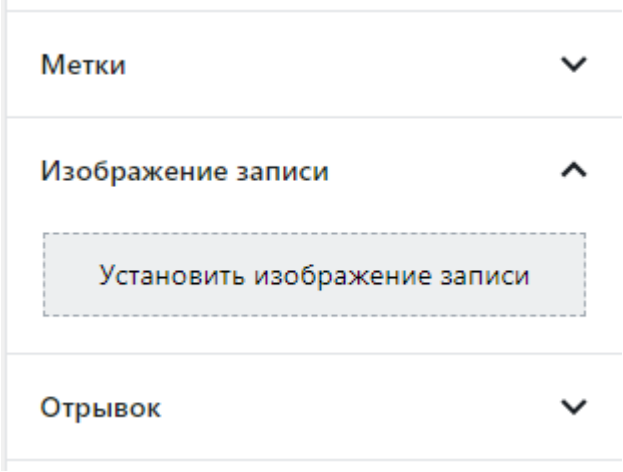


Рисунок 7.5. – Возможность добавления изображения записи при редактировании или создании новой записи в консоли администратора

Произведите добавление файла «m-farmerboy.jpg» идущего в составе архива используемой вёрстки и произведите обновление редактируемой записи. К сожалению добавленное изображение не отобразиться на страничке сайта, а для его вывода мы должны вызвать его на странице «single.php».

За вызов изображения записи на странице отвечает функция «the_post_thumbnail('thumbnail')», добавьте её в файле «single.php» в месте вывода изображения записи (рисунок 7.6) и обновите страницу.

```
</header>
<div class="entry-content-media">
 <div class="post-thumb">
  <?php the_post_thumbnail('thumbnail'); ?>
 </div>
</div>
```

Рисунок 7.6 – Вывод изображения записи в файле «single.php»

После обновления страницы, сразу после идущего блока заголовка записи, будет выведено изображение записи, но его размер не соответствует макету вёрстки. Для его изменения, нам либо необходимо изменить её размер в консоли администратора (настройки – медиафайлы – размер миниатюры), либо добавить свой собственный размер миниатюры в разрабатываемой теме. За реализацию данной возможности в WordPress существует функция «add_image_size()» [10]. Данную функцию также следует вызвать в момент активации темы и регистрации меню сайта, учитывая при этом, что требуемый нам размер миниатюры 615px по ширине и 410px по высоте, имя создаваемой миниатюры будет «post_thumb», а параметр «true» будет отвечать за принудительную обрезку изображения до указанных размеров (рисунок 7.7).

```
}
function register_main_menu() {
 register_nav_menu( 'main_menu', 'Верхнее меню сайта' );
 register_nav_menu( 'footer_menu', 'Меню в подвале сайта' );
 register_nav_menu( 'social_menu', 'Меню социальных сетей' );
 add_theme_support( 'post-thumbnails', array('post') );
 add_image_size( 'post_thumb', 615, 410, true);
}
```

Рисунок 7.7 – Регистрация новой миниатюры с заданным именем и размерами

После регистрации новой миниатюры, необходимо изменить имя вызываемой миниатюры в файле «single.php» (рисунок 7.8) и после обновления

[10] Подробнее о работе функции можно узнать по ссылке: https://wp-kama.ru/function/add_image_size

страницы отображение миниатюра будет соответствовать первоначальной разметке.

```
<div class="entry-content-media">
 <div class="post-thumb">
  <?php the_post_thumbnail('post_thumb'); ?>
 </div>
</div>
```

Рисунок 7.8 – Вызов изображения записи в файле «single.php» с именем миниатюры «post_thumb»

Следующим по вёрстке идет блок контента, на данный момент в нем расположены три параграфа текста, удалите их заменив использованием функции «the_content» (рисунок 7.9).

```
</div>
<div class="entry-content">
 <?php the_content(); ?>
</div>
<p class="tags">
```

Рисунок 7.9 – Вывод основного контента поста на страниц «single.php»

После обновления страницы в браузере на ней должен отобразиться основной контент поста.

Далее по вёрстке идет вывод тегов поста, за их вывод в WordPress отвечает функция «the_tags»[11]. Для автоматического вывода тегов поста, замените код параграфа с классом «tags» на вызов функции «the_tags», так как это показано на рисунке 7.10.

[11] Подробнее о работе функции можно узнать по ссылке: https://wp-kama.ru/function/the_tags

```
 <?php the_content(); ?>
</div>
<p class="tags">
 <?php the_tags('<b>Теги поста:</b> ', ' / '); ?>
</p>

        <ul class="post-nav group">
```

Рисунок 7.10 – Вывод тегов на страницу поста «single.php»

Последним элементом, идущим в составе единичного поста, является навигация к предыдущему и следующему постам, за такую навигацию в WordPress отвечает функция «the_post_navigation»[12]. В нашем случае требуется немного изменить стандартный вывод таких ссылок, для простоты выполнения скопируйте по ссылке: https://wp-kama.ru/function/the_post_navigation пример использования функции и произведите доработку кода в соответствии с рисунком 7.11.

```
</p>
<ul class="post-nav group">
<?php
 the_post_navigation( array(
  'prev_text' => '<li class="prev">' .
   '<strong>Предыдущая запись</strong>' .
   '%title</li>',
  'next_text' => '<li class="next">' .
   '<strong>Следующая запись</strong>' .
   '%title</li>'
 ) ); ?>
 </ul>
```

Рисунок 7.11 – Вывод навигации к предыдущему и следующему постам в файле «single.php»

После закрытия маркированного списка «ul» с классами «post-nav» и «group» заканчивается тег «article» а, следовательно, и вывод поста

Ниже идет вывод комментариев и форма для добавления нового комментария, вывод этих элементов мы произведем несколько иначе чем это задумано разработчиком темы.

[12] Подробнее о работе функции можно узнать по ссылке: https://wp-kama.ru/function/the_post_navigation

Перенесите все содержимое блока с идентификатором «comments» в отдельный файл и назовите его «temp.php» (этот код будет необходим при оформлении формы добавления комментариев, и чтобы его не потерять мы переместим его в отдельный документ).

В блоке с идентификатором «comments» мы должны разместить код позволяющий проверить открыта ли вообще на сайте возможность комментирования, и в случае успеха WordPress должен «подгрузить» файл шаблона комментариев (рисунок 7.12). Сам файл мы должны создать в корне разрабатываемой темы и назвать его «comments.php».

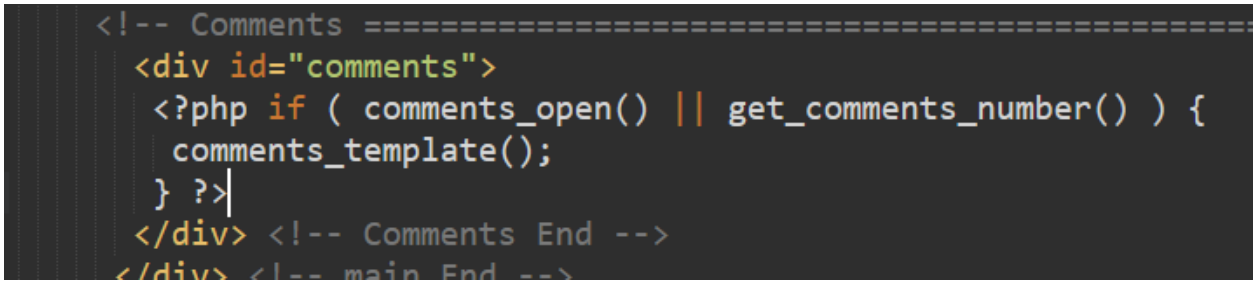


Рисунок 7.12 – Проверка возможности комментирования на сайте

Перед тем как выводить комментарии, нужно удостовериться в их наличии, т.е. произвести проверку, если есть — вывести полный список, если нет — то можно, например, показать пользователю фразу «Комментарии отсутствуют». Так читателю поста будет понятнее, что никто еще ничего не писал, но возможность комментирования существует.

Для осуществления такой проверки необходима реализация конструкции if/else с использованием функция вывода количества комментариев «get_comments_number». В том случае, если функция возвращает 0 (ноль), то выводим «Комментарии отсутствуют», иначе выводим фразу «Комментарии к посту», ну а если оба условия не выполняются, то нам будет выведена фраза «Обсуждения закрыты для данной страницы». Код такого условия, приведен на рисунке 7.13, обратите внимание на то, что код проверки должен быть в файле шаблона вывода комментариев «comments.php».

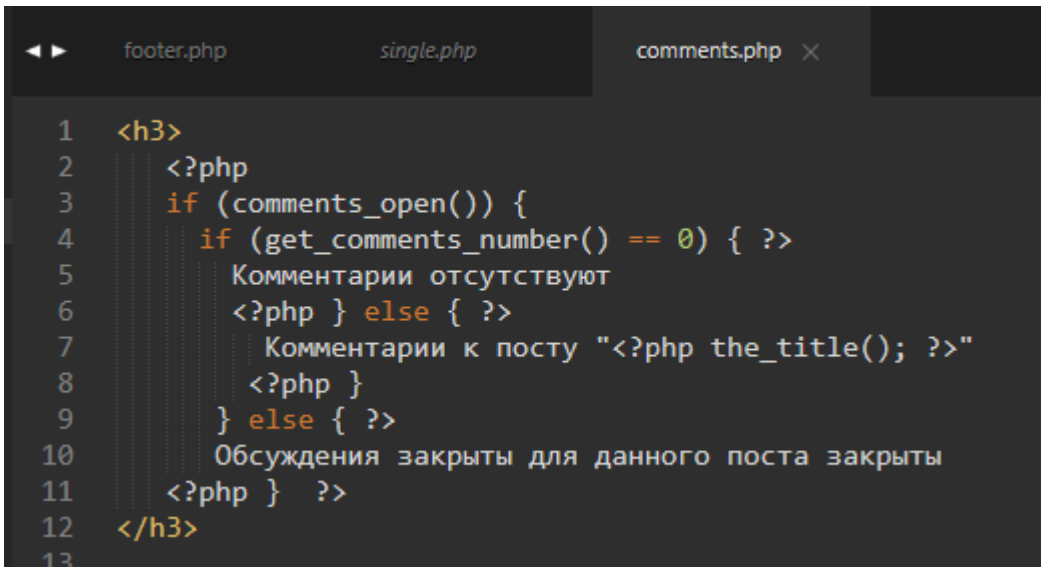


Рисунок 7.13 – Проверка возможности комментирования, количества уже оставленных комментариев к посту в файле «comments.php»

На данном этапе разработки шаблона комментариев, будет выведен заголовок в зависимости от настроек CMS WordPress и оставленных комментариев, самого же вывода комментариев нет. Для того, чтобы вывести комментарии на нашей странице при помощи шаблона комментариев, нам необходимо воспользоваться возможностями функции «wp_list_comments()»[13].

Сама по себе функция отвечает за вывод комментариев, оставленных к постам или страницам и может принимать ряд параметров, в нашем же случае нужны параметр «style» и «avatar_size». Все элементы массива комментариев будут обрамлены в нумерованный список «ol» с классом «commentlist», так как это соответствует нашей верстке (проверьте в файле temp.php), а размер аватара (судя всё по той же верстке) 50*50 пикселей. Сам код функции должен быть помещен в условии, проверяющем наличие комментариев, в случае его успешного выполнения (см. рисунок 7.14).

[13] Более подробно о работе функции можно узнать по ссылке: https://wp-kama.ru/function/wp_list_comments

```
        } else { ?>
        Обсуждения закрыты для данного поста закрыты
    <?php }  ?>
</h3>

<?php if ( have_comments() ) : ?>

<ol class="commentlist">
 <?php
  wp_list_comments( array(
   'style'       => 'ol',
   'avatar_size' => 50,
  ) );
 ?>
</ol>
<?php endif; ?>
```

Рисунок 7.14 – Вывод комментариев в файле шаблона

Последнее, что нам остается сделать, это вывести форму для добавления нового комментария, для выполнения этой задачи в WordPress существует функция «comment_form()»[14]. Эту функцию необходимо вызвать сразу после вывода комментариев в файле «comments.php» (см. рисунок 7.15).

```
   'avatar_size' => 50,
  ) );
 ?>
</ol>
<?php endif; ?>

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

Рисунок 7.15 – Вывод формы добавления комментариев в файле «comments.php»

Данная функция выведет форму «по умолчанию», но для неё существует пример использования, скопируйте этот пример и самостоятельно, обращая внимание на файл «temp.php» произведите адаптацию этого примера с использованием тегов, классов и идентификаторов, используемых ранее в вёрстке.

[14] Более подробно о работе функции можно узнать по ссылке: https://wp-kama.ru/function/comment_form

Кроме того, Вам понадобится изменить порядок полей формы, для этого отлично может подойти функция фильтрации и пересортировки содержимого которая изображена на рисунке 7.16[15]. Разместить её необходимо в том же файле «comments.php» перед вызовом формы комментирования.

```
 ) );
?>
</ol>
<?php endif; ?>

<?php
add_filter('comment_form_fields', 'kama_reorder_comment_fields' );
function kama_reorder_comment_fields( $fields ){
 // die(print_r( $fields )); // посмотрим какие поля есть

 $new_fields = array(); // сюда соберем поля в новом порядке

 $myorder = array('author','email','url','comment'); // нужный порядок

 foreach( $myorder as $key ){
  $new_fields[ $key ] = $fields[ $key ];
  unset( $fields[ $key ] );
 }

 // если остались еще какие-то поля добавим их в конец
 if( $fields )
  foreach( $fields as $key => $val )
   $new_fields[ $key ] = $val;

 return $new_fields;
}

$defaults = array(
 'fields'               => array(
        'author' => '<div class="group">' . '<label for="author">' . __(
```

Рисунок 7.16 – Фильтрация и переопределение полей ввода в форме комментариев

После проведенных действий по приведению внешнего вида формы в соответствии с макетом удалите временный файл «temp.php» и проверьте работоспособность возможности оставлять комментарии к записям.

За вывод отдельных страниц сайта в теме WordPress, как уже и говорилось, отвечает файл «page.php». По своему содержанию он мало чем отличается от файла «single.php», скопируйте этот файл и переименуйте его в файл «page.php». Откройте файл для редактирования и удалите в нём следующие его части:

1. Весь блок с классом «entry-content-media», т.к. вывод изображения страницы как это было в случае записи нам не нужен;

[15] Пример использования можно скопировать по адресу: https://wp-kama.ru/function/comment_form

2. Весь параграф с классом «tags», т.к. вывод тегов на странице тоже не требуется;

3. Маркированный список с классами «post-nav group», т.к. навигация между предыдущими и следующими страницами не нужен;

4. Блок с идентификатором «comments», т.к. комментирование страниц не необходимо;

5. В маркированном списке, находящемся в заголовке содержимого страницы, удалите элемент списка «li» с классами «category tag» и идущий перед ним элемент «span», т.к. вывод рубрики и тегов страницы также не нужен.

На этом, правка данного файла завершена, а в теме присутствует полноценный файл для вывода содержимого страницы.

Произведите действия по созданию файла, отвечающего за вывод отдельной записи (single.php). Не забудьте реализовать функциональные возможности добавления изображения записи. В случае необходимости произведите дополнительную стилизацию элементов. Выведите на странице записи комментарии и форму комментирования. Добейтесь полного соответствия с макетом верстки. По аналогии с действиями, выполняемыми в практической работе выполните создание файла page.php.

Контрольные вопросы:

1. Каким образом возможно изменить порядок вывода полей формы комментирования?
2. Каким образом возможно переопределить используемые в форме комментирования классы и идентификаторы формы комментирования?
3. К каким элементам сайта, находящегося под управлением CMS WordPress возможны комментарии, а к каким нет?

Практическая работа № 8

Шаблоны вывода страниц и записей в теме полученных в результате поиска и отсортированных в соответствии с метками

Цели работы:

- Сформировать представление о назначении отдельных файлов темы, отвечающих за вывод информации;
- Научиться выводить результаты поиска на отдельную страницу;
- Научиться выводить разбиение потока постов используя пагинацию;
- Научиться осуществлять вывод постов в соответствии с фильтрацией по меткам.

Оборудование и материалы:

- Персональный компьютер с установленной операционной системой семейства Windows;
- Локальный web-сервер OpenServer;
- Редактор кода SubLime Text 3;
- Сверстанный макет сайта.

Указания к выполнению

Часто возникает ситуация необходимости организации поиска на страницах сайта и отображения результата. В рассматриваемом случае поиск уже организован при помощи виджета опубликованного в сайдбаре, а вот для вывода результатов поиска мы должны создать отдельную страницу с именем «search.php». За основу файла можно взять файл «index.php», из которого необходимо будет удалить содержимое блока с идентификатором «main» и классами «eight columns» (рисунок 8.1).

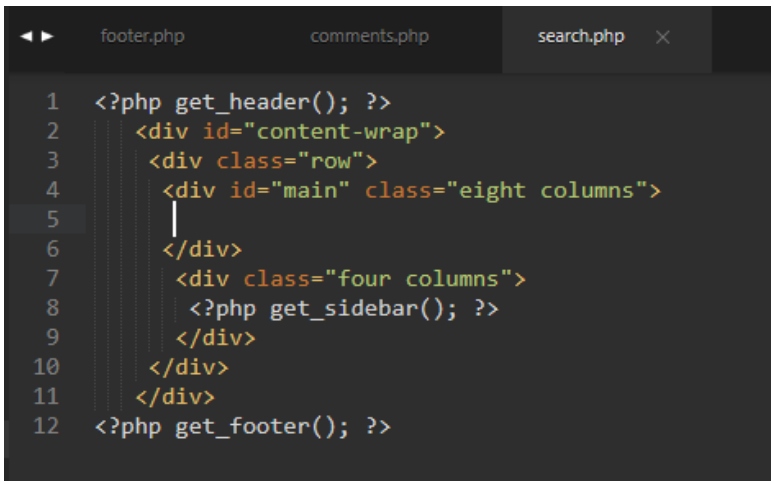


Рисунок 8.1 – Исходный код файла «search.php»

Для начала в данном блоке разместим статическую информацию: в теге заголовка третьего уровня – фразу «Результаты поиска:». После этого необходимо осуществить поиск любой текстовой фразы по сайту и проследить отображаемую информацию. В результате должна отобразится только что впечатанная фраза, следовательно, файл сработал именно как файл, отображающий информацию результатов поиска.

На следующем этапе в файле «search.php» необходимо запустить стандартный цикл WordPress, чтобы вывести найденные в результате поиска данные (рисунок 8.2).

Рисунок 8.2 – Вывод результатов поиска в файле «search.php» при помощи стандартного цикла WordPress

Также, нам необходимо предусмотреть вероятность того, что по поисковому запросу ничего не будет найдено. Для этого необходимо запустить проверку при помощи условия «if» и в случае если результаты поиска есть они выведутся при помощи цикла WordPress, если же результатом условия будет «ложь» в заголовке четвертого уровня выведется фраза «Извините по Вашему запросу ничего не найдено» (Рисунок 8.3).

```
<?php get_header(); ?>
    <div id="content-wrap">
     <div class="row">
      <div id="main" class="eight columns">
       <h3>Результаты поиска:</h3>
       <?php
         if (have_posts()) :
             while (have_posts()) : the_post(); ?>
                 <?php the_title() ?>
             <?php endwhile; ?>
         <?php
         else :
             echo '<h4>Извините по Вашему запросу ничего не найдено</h4';
         endif;
         ?>
      </div>
       <div class="four columns">
        <?php get_sidebar(); ?>
       </div>
     </div>
    </div>
<?php get_footer(); ?>
```

Рисунок 8.3 – Проверка наличия результатов поиска в файле «search.php»

Проверьте корректность работы вывода результатов поиска (сначала попробуйте найти то, что на сайте точно присутствует, а затем то, что точно отсутствует). В случае удачного поиска на странице должны быть выведены заголовки постов где встречается поисковая фраза.

Для более понятного восприятия, а также в целях обеспечения удобства навигации требуется дополнительно вывести с заголовком поста его автора, дату создания, рубрику к которой он относится и вступительный текст поста. Кроме того, мета данные поста (автор, дата публикации, рубрика записи) должны быть ссылками для перехода к соответствующим разделам. Также размер выводимого вступительного текста поста (по умолчанию первые 55 слов поста) мы ограничим первыми 20 словами для удобства восприятия представленной

информации, и введем пагинацию по страницам вывода результатов для случая, когда количество результатов будет больше 10.

Еще готовый код вывода поста со всеми указанными элементами уже есть в нашем распоряжении, в файле «index.php» это «кусок кода» заключенного между тегами «article», скопируйте этот код и вставьте вместо кода вызывающего заголовок поста (рисунок 8.4).

```
while (have_posts()) : the_post(); ?>
 <article class="entry">
  <header class="entry-header">
   <h2 class="entry-title">
    <a href="<?php the_permalink(); ?>" title="Ссылка на: <?php the_title_attribute(); ?>">
     <?php the_title(); ?>
    </a>
   </h2>
   <div class="entry-meta">
    <ul>
     <li><?php the_time('j F Y') ?></li>
     <span class="meta-sep">&bull;</span>
     <li><a href="#" title="" rel="category tag"><?php the_category(', '); ?></a></li>
     <span class="meta-sep">&bull;</span>
     <li><?php the_author_posts_link() ?></li>
    </ul>
   </div>
  </header>
  <div class="entry-content">
   <?php  the_excerpt(); ?>
  </div>
 </article>
<?php endwhile; ?>
```

Рисунок 8.4 – Вывод поста в файле «search.php»

Чтобы осуществить реализацию навигации по страницам результатов поиска, мы должны добавить после закрытия цикла WordPress код вызова пагинации <?php the_posts_pagination(); ?> (рисунок 8.5).

```
        <?php endwhile;?>
        <?php  the_posts_pagination(); ?>
<?php
else :
```

Рисунок 8.5 – Вывод пагинации по страницам результатов поиска

Последнее, что нам осталось сделать это сократить текст вступительной части поста с 55 слов до 20, а также сделать после этого текста ссылку на прочтение всего поста. Это мы реализуем при помощи добавления двух фильтров, первый ограничит количество выводимых слов, а второй добавит

ссылку на полное содержимое. Вставить добавление этих фильтров необходимо перед запуском цикла WordPress (рисунок 8.6).

```
<div id="main" class="eight columns">
 <h3>Результаты поиска:</h3>
 <?php
  add_filter( 'excerpt_length', function(){
    return 20;
  } );
  add_filter( 'excerpt_more', 'new_excerpt_more' );
   function new_excerpt_more( $more ){
   global $post;
   return ' <a href="'. get_permalink($post) . '"> Подробнее...</a>';}
   ?>
 <?php
   if (have_posts()) :
       while (have_posts()) : the_post(); ?>
```

Рисунок 8.6 – Фильтрация вывода вступительного текста поста

На этом правку страницы вывода результатов поиска можно считать законченной, на своё усмотрение Вы можете удалить или добавить некоторые данные к выводу на страницу, добавить элементам какие-либо собственные классы и стилизовать их.

Следующей страницей для разработки станет страница, отображающая записи и страницы в зависимости от выбранного тега или метки, данный файл должен иметь имя «tag.php». Скопируйте файл «category.php», переименуйте его в файл «tag.php» и откройте для редактирования.

Непосредственно сразу после открытия блока с идентификатором «main» и классами «eight columns» разместите заголовок третьего уровня с фразой «Для тега отобраны следующие записи и страницы сайта» Рисунок 8.7.

```
<div id="content-wrap">
 <div class="row">
  <div id="main" class="eight columns">
   <h3>Для тега отобраны следующие записи и страницы сайта</h3>
   <?php if(have_posts()) {while (have_posts()){ the_post(); ?>
    <article class="entry">
     <header class="entry-header">
      <h2 class="entry-title">
```

Рисунок 8.7 – Код вывода заголовка страницы в файле «tag.php»

Далее измените фразу заголовка, добавив в нее функцию «single_tag_title()», которая выводит заголовок текущей метки, а также добавьте после заголовка тег «<hr>», выводящий горизонтальную черту сразу после заголовка (рисунок 8.8).

```
<div class= row >
 <div id="main" class="eight columns">
  <h3>Для тега "<?php single_tag_title() ?>" отобраны следующие записи и стра
  <?php if(have_posts()) {while (have_posts()){ the_post(); ?>
```

Рисунок 8.8 – Вывод заголовка текущей метки в заголовке страницы

Теперь, если в виджете выводящем на сайте метки к постам и страницам, выбрать ту или иную метку, откроется страница на которой будут представлены посты, соответствующие данной метке.

Самостоятельно произведите сокращение выводимой части вступительного текста, проверьте разбиение по страницам в случае большого числа постов, имеющих одну метку (пагинация), отбор необходимых данных для вывода на страницу и их стилизацию. При стилизации обратите внимание на необходимость соответствия собственно выполненной стилизации общей стилизации элементов на сайте.

Произведите действия по созданию файла, отвечающего за вывод результатов поиска по сайту и вывода записей по меткам. В случае необходимости (например, отсутствие стилизованного файла в макете верстки) произведите дополнительную стилизацию элементов.

Контрольные вопросы:

1. Какой файл темы WordPress отвечает за вывод результатов поиска по сайту?
2. При помощи какой функции возможен вывод не всего контента страницы, а только лишь вступительного текста?

Практическая работа № 9

Работа с формами обратной связи

Цели работы:

- Сформировать представление о назначении форм обратной связи на сайте;
- Научиться устанавливать и настраивать плагины для CMS WordPress;
- Научиться настраивать поля форм для использования на сайте;
- Научиться настраивать учетную запись электронной почты для сайта, находящегося под управлением CMS WordPress;
- Научиться вызывать формы обратной связи в всплывающих дочерних окнах.

Оборудование и материалы:

- Персональный компьютер с установленной операционной системой семейства Windows;
- Локальный web-сервер OpenServer;
- Редактор кода SubLime Text 3;
- Сверстанный макет сайта.

Указания к выполнению

В настоящее время практически не один сайт не обходится без наличия каких-либо форм для заполнения пользователем. Примерами использования форм могут быть и формы обратной связи и заказа обратного звонка. Наибольший интерес представляют разнообразные реализации всплывающих форм на сайте. В нашем проекте мы реализуем и форму обратной связи для сайта и эту же форму в виде всплывающего окна.

9.1 Предварительная подготовка к реализации функциональных возможностей

Для того, чтобы дынные из форм не были утеряны, а были пересланы посредством электронной почты администратору сайта, необходимо наличие некоторых плагинов, имеющих соответствующие настройки.

Для управления формами обратной связи на сайтах, находящихся под управлением CMS Wordpress, очень большую популярность и широкое признание пользователей получил плагин - Contact Form 7. Откройте консоль администратора сайта и перейдите в ней к разделу «Плагины», затем нажмите кнопку «Добавить новый» и в появившемся окне произведите поиск соответствующего плагина. На рисунке 9.1 представлен внешний вид необходимого плагина выданного системой поиска.

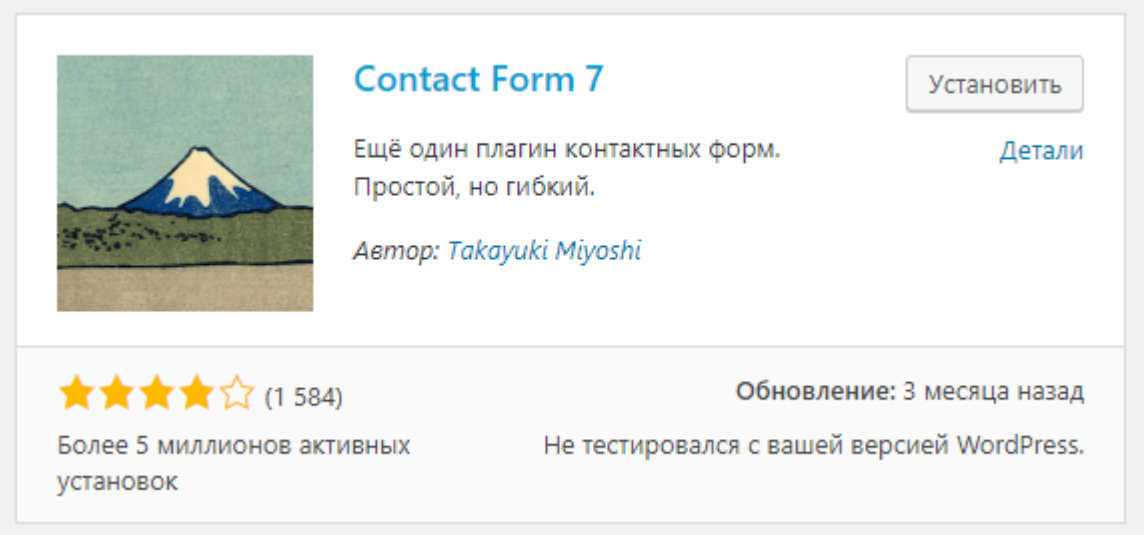


Рисунок 9.1 – Внешний вид карточки плагина Contact Form 7 в репозитории плагинов WordPress

Произведите установку и активацию данного плагина. После установки и активации плагина в системном меню консоли администратора появится соответствующий новый пункт «ContactForm 7» в котором имеется возможность управления уже существующими формами, добавления новых, а также настройки интеграции форм со сторонними сервисами, защищающими формы сайта от заполнения их роботами (см. рисунок 9.2).

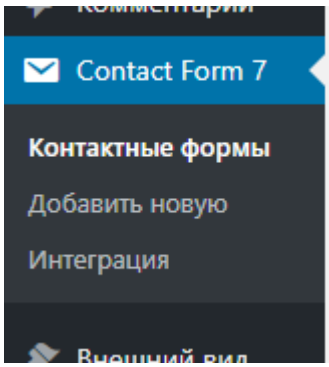


Рисунок 9.2 – Внешний вид пункта меню «Contact Form 7» в консоли администратора

Перейдите к пункту «Контактные формы», по умолчанию, после установки плагина, создается и одна контактная форма – её имя «Контактная форма 1». Для того, чтобы добавить это форму для отображения на сайте необходимо скопировать «шорткод» (см. рисунок 9.3) вставки формы и перейти к редактированию имеющихся на сайте страниц.

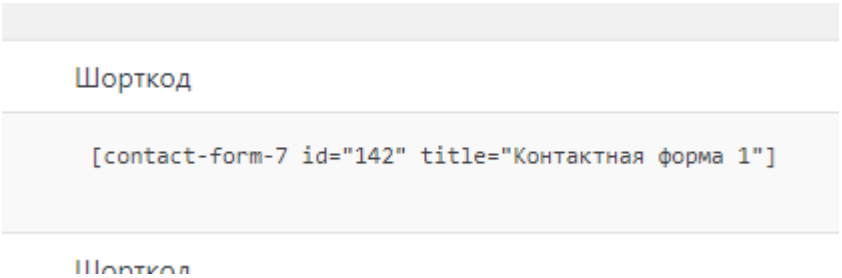


Рисунок 9.3 – Шорткод вставки формы «Контактная форма 1»

Откройте для редактирования страницу «About» и в качестве контента страницы вставьте скопированный ранее шорткод. После этого перейдите к сайту и откройте отредактированную только-что страницу «About», на странице должна отобразиться форма, представленная на рисунке 9.4.

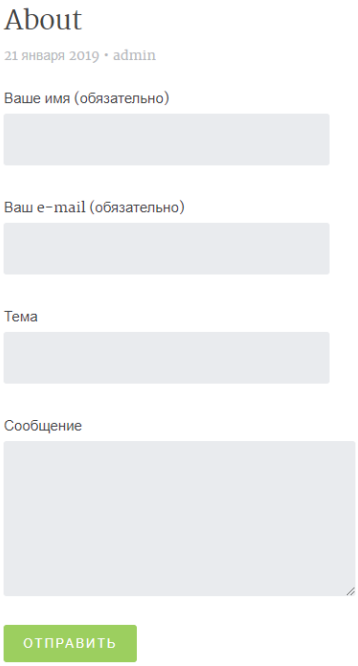


Рисунок 9.4 – Внешний вид формы добавленной на сайт при помощи плагина «Contact Form 7»

Как видно, в форме представлены такие поля как:

- Ваше имя;
- Ваш e-mail;
- Тема;
- Сообщение.

Состав полей формы можно изменить в любое время для этого необходимо перейти к редактированию самой формы в соответствующем меню плагина (см. рисунок 9.5).

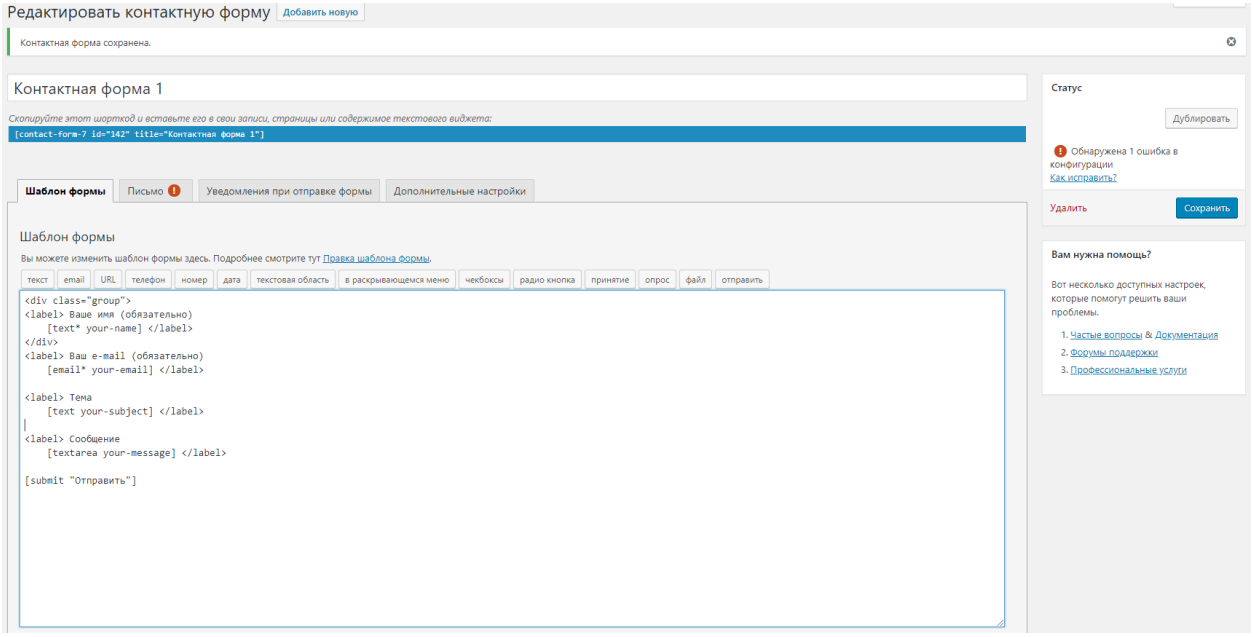


Рисунок 9.5 – Внешний вид редактора форм плагина «Contact Form 7»

Добавьте к контактной форме чекбокс, при этом не забудьте отметить в настройках что это обязательное поле, в качестве параметров добавьте текст «Согласен на передачу моих данных» (этот текст отобразится рядом с чекбоксом), а в качестве «Атрибута id» укажите «accept» (см. рисунок 9.6).

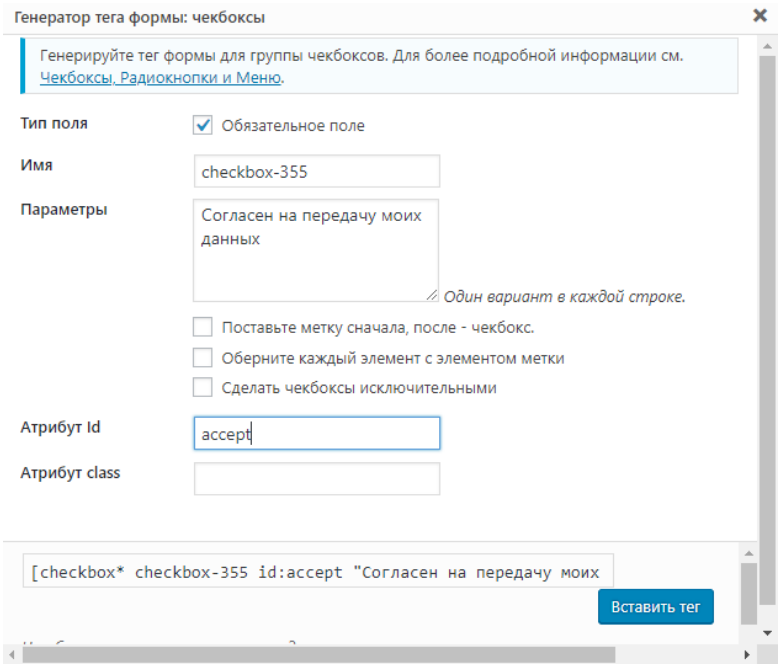


Рисунок 9.6 – Окно генератора тэгов формы плагина «Contact Form 7»

Сам код шаблона формы должен принять вид, изображенный на рисунке 9.7.

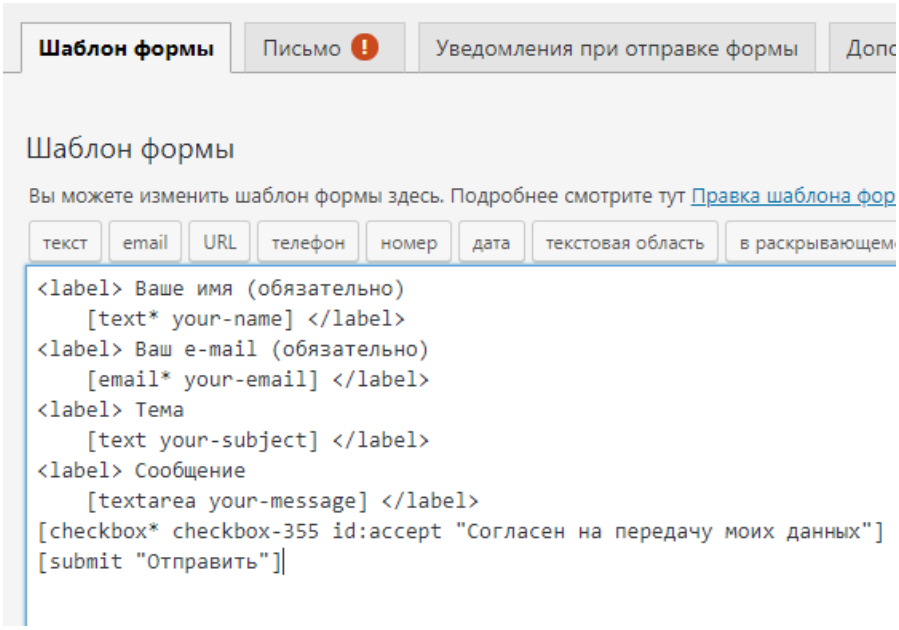


Рисунок 9.7 – Код шаблона формы после редактирования

После редактирования кода формы, сохраните её и перейдите к её просмотру на сайте. К ранее имеющимся полям формы был добавлен новый элемент. обратите внимание на то, что, не отметив имеющийся чекбокс невозможно будет отправить данные формы.

На данном этапе если произвести заполнение полей и попробовать отправить данные формы, они не смогут быть доставлены, так как не настроена отправка почты с сайта. Для этого необходимо установить и настроить еще один плагин с именем «Easy WP SMTP»

9.2 Настройка отправки почтовых сообщений с сайта находящегося под управлением WordPress

Перейдите к разделу консоли администратора «Плагины», нажмите кнопку «Добавить новый» произведите поиск, установку и активацию плагина «Easy WP SMTP». Внешний вид необходимого плагина выданного системой поиска представлен на рисунке 9.8.

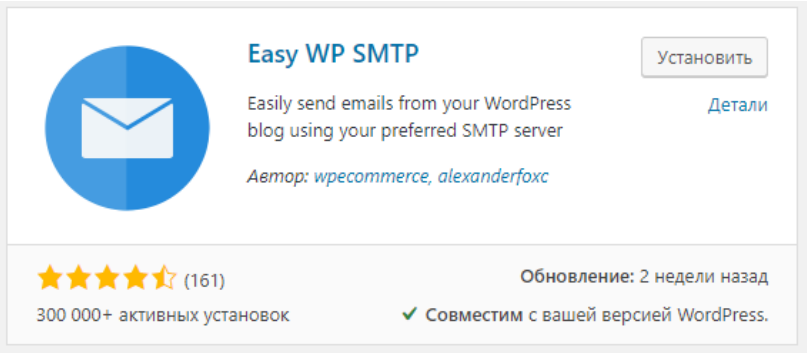


Рисунок 9.8 – Внешний вид карточки плагина Easy WP SMTP в репозитории плагинов WordPress

После установки и активации плагина система сама перенаправит Вас в окно с уже установленными плагинами, а вверху страницы будет представлено системное сообщение о необходимости настройки учетных данных сервера исходящей почты SMTP в меню настроек (см. рисунок 9.9). Перейдите к данному меню.

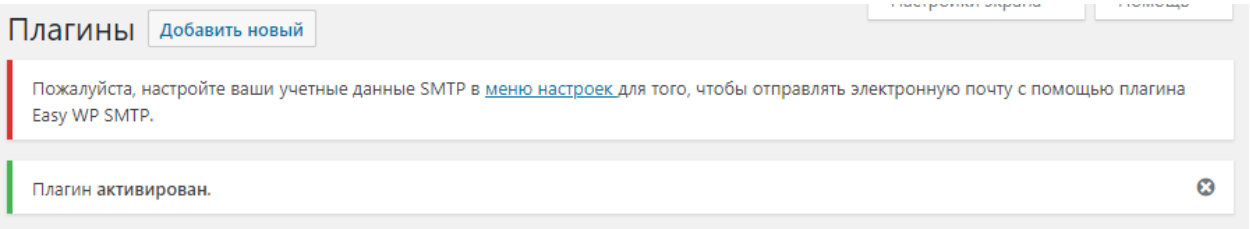


Рисунок 9.9 – Системное сообщение о необходимости настройки учетных данных

Обратите внимание на то, что для корректной работы. потребуются актуальные и реально существующие данные учетной записи любого почтового сервиса (Yandex, Gmail, Mail.ru или другой).

Конкретные настройки могут отличаться в зависимости от сервиса, а необходимым будет внесение настроек в следующие поля:

- От кого (Адрес электронной почты)
- От кого (Имя)
- SMTP-хост
- Тип шифрования
- Порт SMTP
- Имя пользователя SMTP

- Пароль SMTP

В моем случае (использование почтового сервиса Yandex) настройка принимают следующие значения:

От кого (Адрес электронной почты)	doroh83@yandex.ru
От кого (Имя)	Учебный сайт на WordPress
SMTP-хост	smtp.yandex.ru
Тип шифрования	SSL/TLS
Порт SMTP	465
Имя пользователя SMTP	doroh83
Пароль SMTP	Мой персональный пароль ********

После проведенных настроек перейдите на вкладку «Проверочное письмо», заполните соответствующие поля, и отследите успешность отправки письма. В случае правильной настройки и успешности отправки письма, данные ранее созданной формы на странице «About» также должны успешно передаваться с сайта на адрес электронной почты указанной в настройках плагина Easy WP SMTP. Проверьте Отправку данных формы сайта.

9.3 Всплывающие формы

Современным является решение реализации форм в всплывающих окнах. Для обеспечения функционала всплывающих окон на сайте под управлением WordPress необходимо установить один из соответствующих плагинов, например, Easy FancyBox. Произведите поиск, установку и активацию данного плагина из репозитория плагинов WordPress.

Внешний вид необходимого плагина выданного системой поиска представлен на рисунке 9.10.

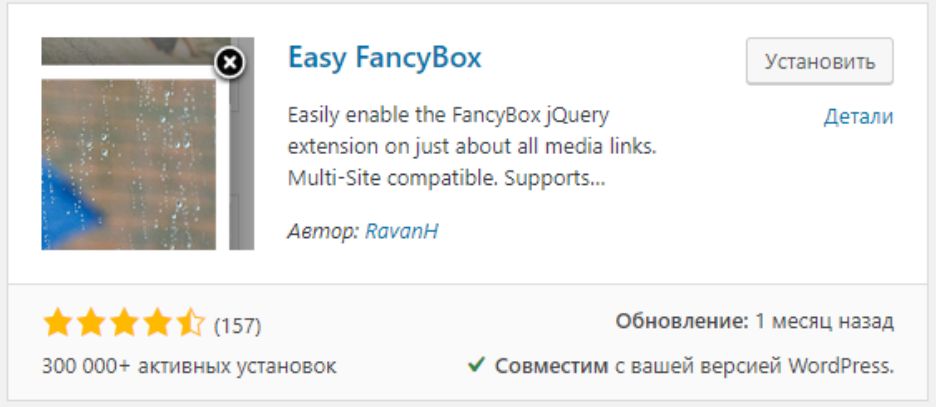


Рисунок 9.10 – Внешний вид карточки плагина Easy FancyBox в репозитории плагинов WordPress

Плагин Easy FancyBox требует минимальной настройки для того, чтобы он мог отображать данные форм. Сразу после установки плагина Вы будете перенаправлены к разделу управления плагинами, найдите в перечне плагинов плагин Easy FancyBox и перейдите к его настройкам. В настройках плагина отметьте в разделе медиа поддержку отображения «Инлайнового контента» и сохраните изменения (см. рисунок 9.11).

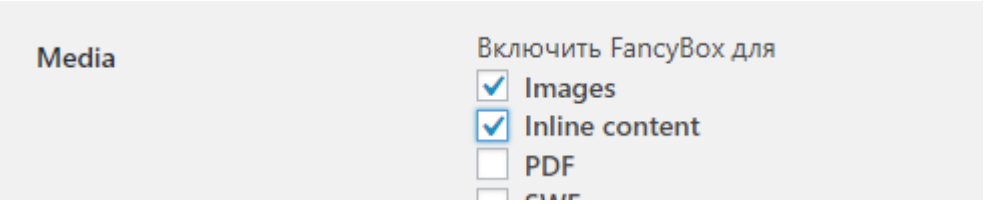


Рисунок 9.11 – Поддерживаемые форматы отображения данных плагином Easy FancyBox

На следующем этапе необходимо создать HTML-элемент, который будет вызывать требуемую форму. Перейдите к подразделу «Виджеты» раздела «Внешний вид» и произведите добавление нового виджета в состав уже имеющихся на сайте. Тип нового виджета будет «Текст», а разместить его необходимо первым в списке (это будет более удобно для будущего пользователя, так как кнопка вверху сайта будет привлекать внимание сразу после открытия сайта в браузере). В создаваемом виджете обязательно необходимо переключить редактор в режим «Текст» и добавить код (см. рисунок 9.12) вызова всплывающего окна. После редактирования виджета произведите

его сохранение и перейдите к сайту, обновите его и протестируйте новые возможности.

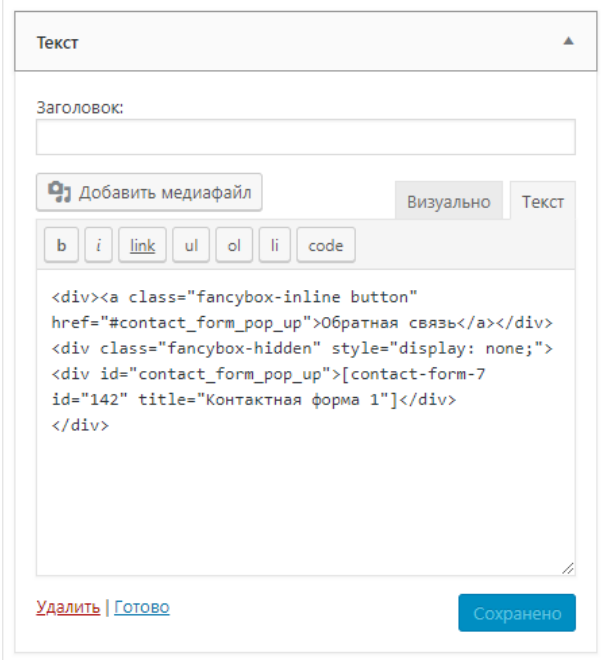


Рисунок 9.12 – Код вызова всплывающего окна в создаваемом виджете

На рисунке 9.13 изображена форма обратной связи, размещенная во всплывающем окне.

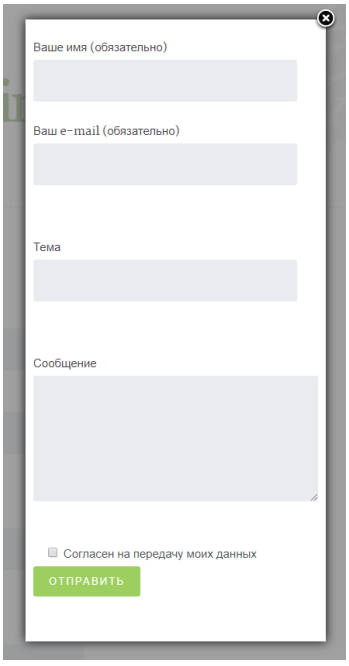


Рисунок 9.13 – Внешний вид формы вызванной в всплывающем окне

Данная форма в настоящий момент выглядит не совсем в соблюдении стилевого оформления сайта, отдельные поля ввода сильно отступают от идущих до них полей. Для устранения этого недостатка отредактируйте контактную форму в настройках плагина «ContactForm 7» так как это изображено на рисунке 9.14.

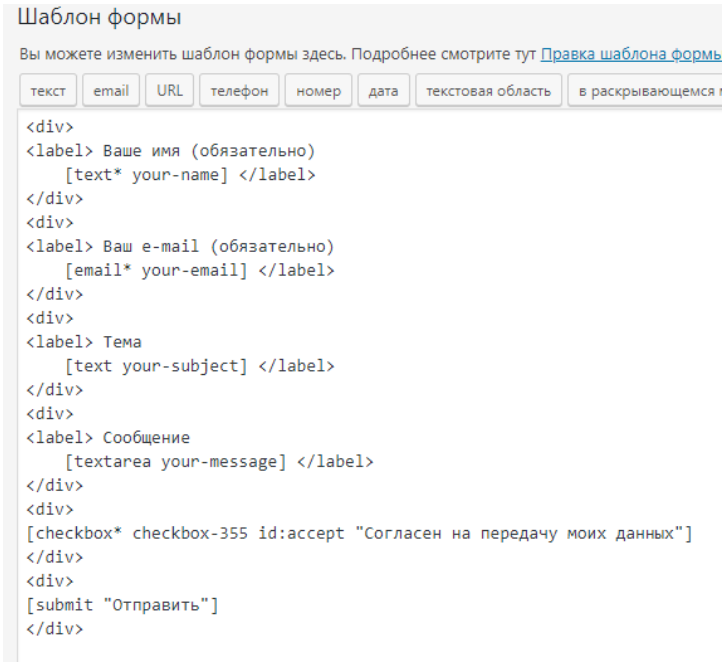


Рисунок 9.14 – Код формы обратной связи после внесения изменений

Также потребуется корректировка файла default.css, суть которой заключается в том, что каждому полю для ввода 100% ширину и внешний отступ равный нулю (см. рисунок 9.15).

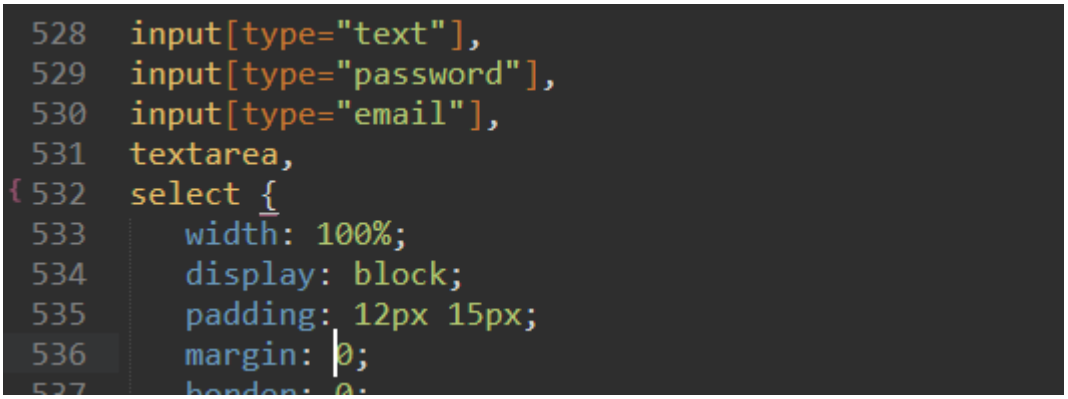


Рисунок 9.15 – Фрагмент кода файла default.css после внесения в него изменений

После этого расстояния между полями ввода примут значения допустимые дизайном сайта, внешний вид формы после редактирования формы и стилевого оформления представлен на рисунке 9.16.

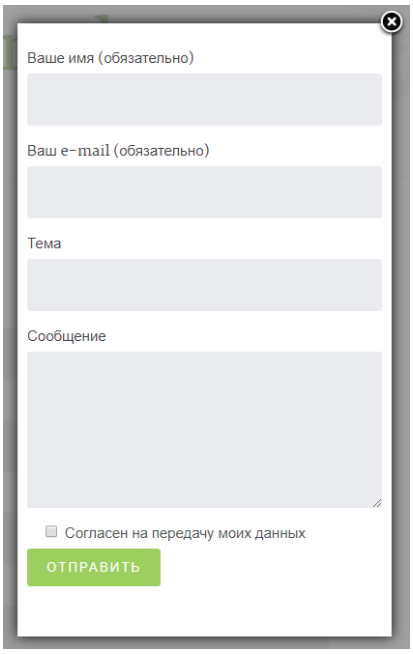


Рисунок 9.16 – Внешний вид формы вызванной в всплывающем окне после редактирования и стилизации

Произведите установку необходимых плагинов и их настройку для реализации работы форм обратной связи как на отдельной странице, так и во всплывающем окне. При необходимости произведите дополнительную стилизацию элементов.

Контрольные вопросы

1. Какой плагин необходим для создания формы обратной связи на сайте под управлением WordPress?
2. Какой плагин осуществляет поддержку работы отправки сообщений с сайта?
3. При помощи какого плагина возможен вызов всплывающего окна?

Практическая работа № 10

Перенос сайта и размещение на хостинге

Цели работы:

- Научиться переносить сайт на другой локальный web-сервер;
- Научиться размещать сайт на внешнем хостинге.

Оборудование и материалы:

- Персональный компьютер с установленной операционной системой семейства Windows;
- Локальный web-сервер OpenServer;
- Редактор кода SubLime Text 3;
- Сайт находящийся под управлением CMS WordPress.

Указания к выполнению

Очень часто в процессе или по окончании разработки сайта, его необходимо разместить на хостинге или перенести на другой локальный web-server. Обе эти ситуации имеют как схожие черты, так и отличия.

10.1. Создание резервной копии сайта, находящегося под управлением CMS WordPress

Для переноса сайта на другой локальный web-сервер необходимо в первую очередь иметь копию всех его файлов. Так как в общем случае копирование большого числа файлов даже незначительного размера, будет очень длительным процессом, необходимо создать архив всех файлов сайта (лучше если это будет архив формата Zip, так как этот формат является наиболее распространенным на хостинговых площадках, а компьютеры, находящиеся под управлением Windows, не требуют установки дополнительного программного обеспечения для работы с архивами данного формата). Создайте такой архив.

На следующем шаге потребуется база данных уже существующего сайта в формате SQL. Для того, чтобы её получить, необходимо воспользоваться инструментом PHPMyAdmin, входящим в состав локального web-сервера OpenServer. Найти данный инструмент можно в подменю «Дополнительно»

меню панели управления OpenServer. Откройте его (имя пользователя root? пароль отсутствует) и перейдите к интересующей базе данных (в случае строгого выполнения данных указаний это база данных с именем wpdb) (см. рисунок 10.1).

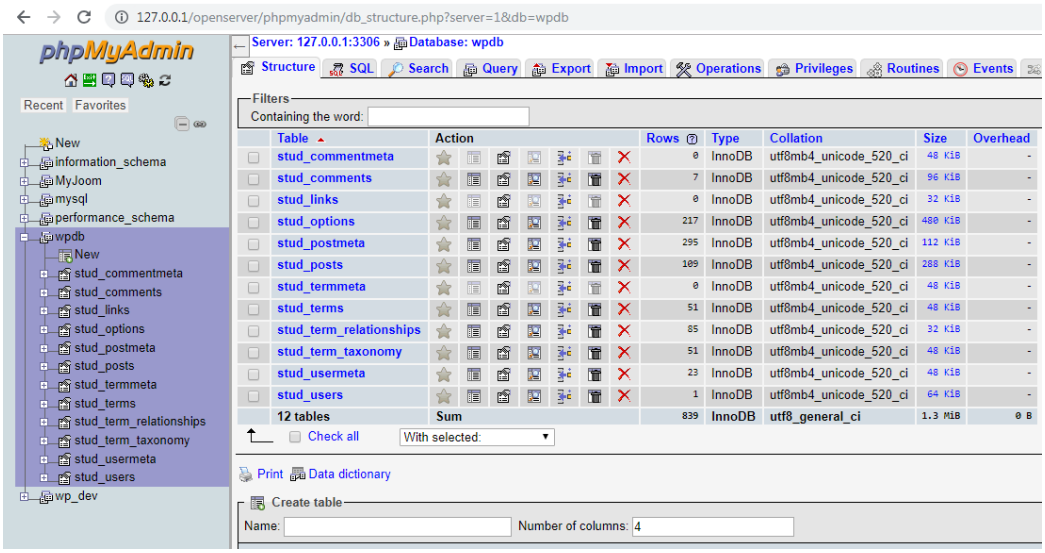


Рисунок 10.1 – База данных с именем wpdb открыта в инструменте PHPMyAdmin

При открытой базе данных перейдите на вкладку «Export/Экспорт» и в новой вкладке нажмите кнопку «Go/Вперед», после этого будет скачана требуемая база данных в формате sql (см. рисунок 10.2).

Рисунок 10.2 – Скачанная база данных в отображаемая в менеджере загрузок браузера Google Chrome

10.2 Перенос сайта, находящегося под управлением CMS WordPress на другой локальный web-сервер

В данном примере будет рассматриваться создание копии уже имеющегося сайта, находящегося под управлением CMS WordPress в другую папку, что будет означать смену его доменного имени.

Создайте в папке «Domains» локального web-сервера OpenServer папку с именем «mysite» и произведите в неё извлечение файлов ранее созданного архива (обратите внимание на то, что все файлы и папки архива должны

располагаться непосредственно в корне созданной папки). Перезагрузите OpenServer и проверьте, что ранее созданная папка отобразилась как новый домен в составе доменов локального сервера.

Поле этого необходимо перенести и базу данных сайта. Для этого также необходимо будет воспользоваться инструментов PHPMyAdmin, откройте его и создайте при его помощи новую базу данных с именем «mysite». После создания базы данных перейдите на вкладку «Import/Импорт», при помощи кнопки «Выдерите файл» вызовите окно выбора файла и укажите в нем расположение ране скачанной базы данных, а затем нажмите кнопку «Go/Вперед». Будет произведен импорт базы данных в указанную базу, но уже с новым именем.

На следующем шаге нам потребуется изменить записи одной из таблиц базы данных, её имя должно заканчиваться на _options (более точного наименования привести невозможно, так как оно зависит от используемого при установке WordPress префикса таблиц базы данных). Откройте указанную таблицу для редактирования и найдите в ней строки «siteurl» и «home». Измените их значения вместо http://wordpress на http://mysite.

Последнее, что требуется изменить, это имя используемой базы данных в конфигурационном файле WordPress с именем «wp-config.php». Откройте данный файл в любом HTML-редакторе и измените имя базы данных с «wpdb» на «mysite» (см. рисунок 10.3).

```
19  */
20
21  // ** Параметры MySQL: Эту информацию мо
22  /** Имя базы данных для WordPress */
23  define('DB_NAME', 'mysite');
24
25  /** Имя пользователя MySQL */
26  define('DB_USER', 'root');
```

Рисунок 10.3 – Фрагмент конфигурационного файла CMS WordPress? касающийся настроек используемой базы данных

После произведенных действий произведите открытие нового сайта в браузере, в случае отсутствия ошибок и правильного отображения страницы работу можно считать успешно выполненной.

10.3 Размещение сайта, находящегося под управлением CMS WordPress на хостинговой площадке

В нашем примере будет рассмотрен вариант размещения сайта на хостинге hostiman расположенного по адресу https://hostiman.ru/.

Пройдите регистрацию на данном сайте, заказав при этом бесплатное доменное имя и бесплатный хостинг, при этом Вам будет предоставлена возможность размещения 2-х сайтов, а дисковое пространство, предоставленное Вам будет равно 1000 Мб.

Перейдите в панель управления хостингом (в моём случае её адрес: https://s1.hostiman.ru:1500/manager/ispmgr), данная панель основана на использовании IPSmanager (довольно распространенной панели для хостинг-провайдеров).

Используя правое меню панели управления перейдите в подраздел «Менеджер файлов» раздела «Главное», а непосредственно в «Менеджере файлов» в папку «WWW» и папку созданного для Вас домена (Рисунок 10.4).

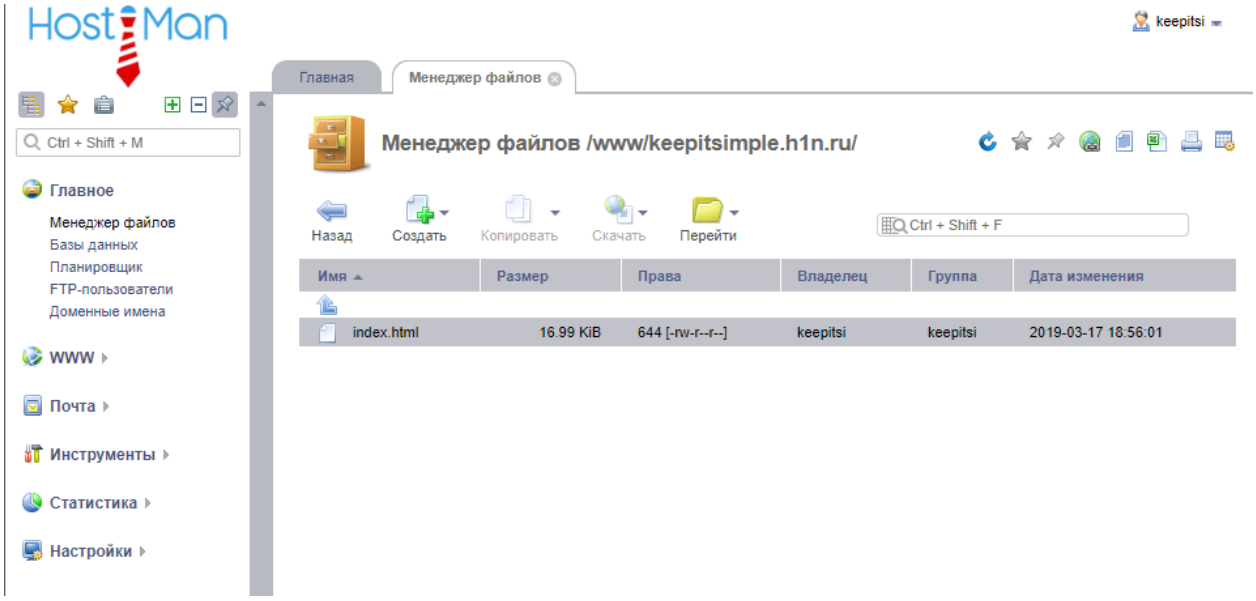


Рисунок 10.4 – Файловый менеджер панели управления хостингом ISPmanager

В дальнейшей работе будут использоваться кнопки менеджера файлов, нажмите кнопку «Закачать», а в открывшейся вкладке произведите выбор файла с архивом сайта, который был ранее создан и нажмите кнопку «Ok». Архив сайта будет закачан на хостинг, и Вы увидите его в списке файлов.

Выделите только-что закачанный файл и нажмите кнопку «Извлечь» а затем «Ok». Архив сайта будет распакован в корень Вашего домена, а сам архив теперь можно будет удалить с хостинга, для этого выделите его и нажмите кнопку «Удалить».

Далее нам потребуется инструмент PHPMyAdmin, найти его можно в разделе «Инструменты» основного меню панели управления хостингом. Откройте его, система предложит создать новую базу данных, пользователя и пароль для доступа к базе данных. Создайте их на своё усмотрение помнив при этом, что использовать необходимо латинские символы, а сами данные будут необходимы в дальнейшем и их необходимо запомнить. Так, например, я создал базу данных с именем my_wpdb.

Еще раз откройте инструмент PHPMyAdmin, который будет открыт в новом окне и пройдите авторизацию используя ранее созданные имя пользователя и пароль. Так же, как и в примере с переносом сайта на другой локальный сервер, произведите импорт базы данных и правку адреса сайта и домашней страницы (в Вашем случае это будет домен созданный для Вас хостинг-провайдером).

Еще раз обратитесь к «Менеджеру файлов», в папке «WWW» удалите файл «index.html», который располагался так сразу после установки и откройте для редактирования файл «wp-config.php» (используя при этом двойной щелчок). Отредактируйте значения параметров Имя базы данных для WordPress, Имя пользователя MySQL, Пароль к базе данных MySQL. Сохраните изменения и откройте сайт в браузере. В случае отсутствия ошибок и правильного отображения страницы работу можно считать успешно выполненной.

Подготовьте файлы архивов сайтов и из баз данных. Произведите регистрацию услуг бесплатного хостинга и разместите Ваши сайты в сети интернет. Результаты как самостоятельной работы, так и практических работ разместите на учебном компьютере с запущенным локальным web-

сервером по указанию преподавателя. Для итоговой проверки работ предоставьте адреса сайтов в сети интернет

Контрольные вопросы:

1. Какой набор файлов необходим для полного переноса сайта, находящегося под управлением CMS WordPress?
2. Обязательное изменение каких параметров строго необходимо при переносе сайта по новому доменному имени?
3. В каком файле хранится информация о параметрах соединения сайта с базой данных?

Рекомендуемая литература

Основные источники

1. Алексей Сергеев - Создание сайтов на основе WordPress. Учебное пособие. – М.: Лань, 2015, 128 с.
2. Трис Хассей WordPress для профессионалов. – М.: Экспо, 2012, 432 с.
3. Б. Уильямс, Д. Дэмстра, Х. Стэрн WordPress для профессионалов. Разработка и дизайн сайтов. – Спб.: Питер, 2014, 464с.
4. Дэрил Бартлетт Wordpress для начинающих. – М.: Эксмо, 2017, 208 с.

Интернет-ресурсы

1. https://wp-kama.ru/- официальный кодекс Wordpress в Российской Федерации;
2. https://www.styleshout.com/ - сайт с размещенными макетами сверстанных сайтов

Программное обеспечение компьютерных сетей и Web-серверов

Практикум

Составители:

С.В. Дорохов, М.В. Дорохова

Формат 60×90 1/16 Бумага офисная. Гарнитура Times New Roman. . Усл.печ.л. 13,875
Уч.-изд.л. 2,79. Тираж 30 экз. Отпечатано с готового оригинал-макета ГБПОУ ВО
«Воронежский государственный профессионально-педагогический колледж»
394016, г. Воронеж, пер. Ученический, 1