СРЕДНЕЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАНИЕ

JAVASCRIPT И PHP

Content management system

В. В. Янцев

ЛАНЬ

E.LANBOOK.COM

В. В. ЯНЦЕВ

JAVASCRIPT И PHP. CONTENT MANAGEMENT SYSTEM

Учебное пособие

ЛАНЬ

· САНКТ-ПЕТЕРБУРГ · МОСКВА · КРАСНОДАР ·
2022

УДК 004
ББК 32.973.4я73

Я 65 **Янцев В. В.** JavaScript и PHP. Content management system : учебное пособие для вузов / В. В. Янцев. — Санкт-Петербург : Лань, 2022. — 192 с. : ил. — Текст : непосредственный.

ISBN 978-5-507-44846-3

Современные веб-ресурсы невозможно представить без систем управления, которые принято называть Content management system или сокращенно CMS. Многие разработчики используют не готовые CMS, например WordPress, Joomla, Drupal, 1С-Битрикс, а пишут собственные. Для опытного программиста разработка подобной системы видится вполне реальной. Для начинающих это серьезный труд. Но если есть желание, то все «подводные камни» удастся преодолеть. А поможет в этом данная книга. В ней рассказывается о механизмах разработки очень простой, но вполне работоспособной CMS, которую можно использовать для создания и управления небольшими сайтами. По сути, это рассказ о первых шагах, с которых начинается освоение такой интересной стези, как написание CMS. Естественно, что, изучив предложенное автором программное обеспечение, вы можете пойти дальше и существенно нарастить его функционал или, используя полученные навыки, разработать собственный вариант системы управления.

Рекомендовано в качестве дополнительной литературы для студентов вузов, обучающихся по направлению «Информатика и вычислительная техника».

Освоить материал книги вам поможет zip-архив с полным набором файлов рассмотренной CMS. Вы сможете запустить систему на своем компьютере. Для этого книга содержит подробные инструкции по созданию локального хостинга на ПК. Дополнительные материалы доступны в электронной библиотечной системе «Лань» по ссылке или QR-коду, указанным ниже.

УДК 004
ББК 32.973.4я73

Обложка
П. И. ПОЛЯКОВА

1. ВВЕДЕНИЕ

Современные веб-ресурсы невозможно представить без систем управления, которые принято называть Content management system (CMS).

Создание профессиональной системы управления — сложная и трудоемкая задача. И тем не менее многие разработчики используют не готовые CMS, а пишут собственные. Иногда популярные системы, такие как WordPress, Joomla, Drupal, 1С-Битрикс и др., оказываются слишком громоздкими для тех или иных проектов. Поэтому рациональнее написать собственную, более простую, «заточенную» под конкретную задачу. Это как с покосом травы на даче. Конечно, можно запустить на участок комбайн, но гораздо проще справиться обычной бензокосилкой.

Автор начал заниматься программированием в 2003 г. С 2004 г. за довольно короткий промежуток времени успел поработать в нескольких студиях веб-дизайна. И вот на что обратил внимание. В те годы почти каждая такая студия, большая или маленькая, старалась написать собственные «движки» для сайтов. Получались они очень разными — от весьма простеньких и неказистых до серьезных профессиональных разработок.

Не остался в стороне от этой моды и автор. Только писал такие системы не во взаимодействии с группой программистов и дизайнеров, а полностью самостоятельно. Назывались они непритязательно — СОУС (Система оперативного управления сайтом), ИКС (Интерактивный конструктор сайтов), КПСС (Конструктор простых социальных сетей). Естественно, автор постоянно их совершенствовал, наращивал функциональные возможности и т. д. До 2020 г. автор и многие его клиенты вполне успешно пользовались данными разработками.

Однако наступил момент, когда автор уже не мог в одиночку конкурировать с такими проектами, как WordPress, Joomla, Drupal, 1С-Битрикс и др.

В итоге все мои системы постепенно были отправлены в архив, после чего автор занялся другими разработками. Но по-прежнему интересовался тем, что происходит в индустрии создания «движков». А самое главное, продолжал заниматься написанием различных компонентов, которые являлись составными частями CMS.

У читателей, наверное, уже появился вопрос: к чему этот исторический экскурс? Он демонстрирует, что на сегодняшний день у автора накопился немалый опыт создания самых разных CMS и их отдельный частей для самых разных веб-проектов. Этим опытом автор и хочет поделиться на страницах данной книги.

1.1. О чем эта книга

Книг о синтаксисе языка программирования JavaScript много. Даже очень много. Конечно, каждая из них по-своему хороша, интересна и полезна, но в конечном счете все они об одном и том же.

Есть ряд книг, в которых излагаются не только основы языка, но и даются примеры сценариев. Они могут объяснить начинающему разработчику, каким образом создаются реальные проекты.

Но изданий, рассказывающих о такой популярной среди программистов теме, как создание CMS, практически нет. Частично автор попытался осветить этот вопрос в своем учебнике «JavaScript. Визуальные редакторы». Но там рассказ велся лишь об одной из составных частей CMS — редакторах WYSIWYG.

Поэтому автор решил написать новую книгу, целиком посвященную теме создания полностью законченной системы управления, которая позволяла бы конструировать сайты с нуля. Специально для данной книги было написано довольно простое ПО, которое и будет рассмотрено в дальнейшем.

Но это будут не просто теоретические рассуждения. Если в книге представлены лишь фрагменты кода, то, скачав по ссылке **https://editjs.ru/CMS.zip** zip-архив, вы получите в свое распоряжение весь проект, включая сам движок и демонстрационный сайт, выполненный в двух вариантах. Вы можете проверить в действии и сайт, и систему управления. А также читать книгу и одновременно изучать «живой» код, записанный в тех или иных файлах.

1.2. Особенности проекта

Автор ставил перед собой задачу сделать проект максимально простым, чтобы разобраться в нем и повторить его мог даже начинающий программист. Поэтому было принято решение отказаться от применения базы данных и использовать вместо нее текстовые файлы. Тем более что проект не содержит ничего такого, что необходимо было скрывать от посторонних лиц. Вся хранящаяся в файлах информация выводится на страницы сайта в открытом виде. Применение БД сделает проект более сложным, более объемным, а его описание неоправданно увеличит объем книги. Больше код — соответственно, больше «вес» архива проекта, а его автор тоже хотел сделать максимально «легким». Освоив материал, читатель самостоятельно сможет заменить текстовые файлы таблицами базы данных.

Итак, проект очень простой. Но в нем есть один компонент, сделанный на профессиональном уровне, — визуальный редактор страниц. Достаточно сказать, что в JavaScript-файле, обеспечивающем все функции редактора, почти 700 строк кода. Но пусть вас это не пугает. В книге приведены подробные, исчерпывающие пояснения ко всем компонентам проекта, включая этот огромный файл.

Редактор не только профессиональный, но и очень удобный. Большинство систем WYSIWYG предоставляет разработчику окно редактирования, размеры которого могут совершенно не совпадать с истинным размером соответствующей части страницы. В результате, то, что хорошо смотрится в WYSIWYG, на реальном сайте может иметь совсем иной вид. И приходится разработчику по новой править страницу. Наш редактор избавлен от такого недостатка. В нем предусмотрен механизм, благодаря которому вы сразу видите, как будет выглядеть страница с внесенными изменениями еще до того, как вы записали их в файл. Более того, вы вносите изменения не в каком-то абстракт-

ном окне, а прямо в самой странице. Благодаря этому требуемый результат достигается с первой попытки.

Еще один важный момент. Как ясно из названия книги, главные действующие лица в ней — языки программирования JavaScript и PHP. Первое место по объему кода занимает JavaScript, соответственно PHP — второе, отставая не на много. Поэтому было бы неплохо, чтобы читатель знал основы этих языков хотя бы на уровне начинающего.

HTML-разметке и таблицам стилей CSS мы тоже уделим внимание, так как их доля в коде проекта заметна. Поэтому быть знакомыми с HTML и CSS читателю также необходимо.

На что еще хотел бы обратить внимание читателя: во избежание терминологических споров по языку JavaScript автор ориентировался на определения, данные на страницах сайта **https://developer.mozilla.org/ru/**.

1.3. Оформление сценариев

Сценарии имеют различное типографское оформление в соответствии с их размещением в тексте.

Если код в книге выделен в отдельный блок, то он оформлен моноширинным шрифтом, например так:

```
function coor(ev)
   {
   let v=ev.pageY;
   let sv=document.getElementById("bpan").offsetTop;
   dv=v-sv;
   addEventListener("mousemove", neco);
   }
```

Если фрагменты сценария внедрены непосредственно в текст, то в этом случае части кода выделены полужирным шрифтом, например, так:

— если условие **ans = 1** ложно.

Обратите внимание: в некоторых блоках программ сделан перенос части кода на вторую и даже на третью или четвертую строку (из-за недостатка ширины страницы в книге). В реальном сценарии код записывается одной строкой. Запомните правило: все переносы строк кода существуют только в их типографском воспроизведении. Если вы в дальнейшем столкнетесь с подобной ситуацией, учитывайте данный аспект.

Еще один момент. Обычно таблицы стилей какой-либо страницы принято записывать следующим образом (пример):

```
#divfl {
   position: absolute;
   left: 0;
   display: flex;
   flex-wrap: wrap;
   z-index: 2;
   width: 100%;
   top: 250px;
}
```

Однако подобный код занимает в книге слишком много места. Поэтому автор применил сокращенную форму записи стилей, например так:

```
#divfl {position: absolute; left: 0; display: flex;
                        flex-wrap: wrap; z-index: 2;
                           width: 100%; top: 250px;}
```

Как видите, экономия налицо: в этом случае вместо девяти строк на запись потрачено только три.

И последнее. В описании программ нередко имеет смысл не приводить весь код, который относится к разбираемому фрагменту, а показывать только ту его часть, что важна в контексте данных объяснений. Поэтому в таких случаях автор вводит сокращение в виде трех точек:

```
...
```

Например, сокращенный вариант для предыдущего примера с настройками стилей может выглядеть так:

```
#divfl {position: absolute; left: 0; ...}
```

Тем самым мы показываем, что на данном этапе описания нам важно обратить внимание исключительно на позиционирование элемента.

2. ПРЕЖДЕ ЧЕМ НАЧАТЬ

До того как мы приступим к написанию CMS, автор посчитал необходимым дать некоторую предварительную информацию:

— подробнее коснуться того, что такое CMS;

— рассказать, какие технологии и методы необходимо будет применить в нашем проекте, в первую очередь в визуальном редакторе;

— пояснить смысл терминов, которые нам придется использовать неоднократно.

Поэтому данный материал я решил выделить пусть и в небольшую, но самостоятельную главу.

2.1. CMS

Представим такую ситуацию: вы разработали сайт, состоящий из нескольких HTML-страниц, и разместили его в Сети. Пройдет немного времени — и вы обнаружите, что информацию на одной или нескольких страницах необходимо отредактировать, а может, и кардинально изменить. Даже самые стабильные интернет-ресурсы нуждаются в периодическом обновлении. А что говорить про сайты, которые рассказывают о быстро меняющихся вещах, например о строительстве жилого микрорайона или о новинках в мире компьютеров. В таких случаях обновление информации происходит чуть ли не каждый день.

И вот вы стоите перед задачей ежедневно редактировать одну или несколько страниц и ставить их вместо устаревших версий. Выглядит это примерно так: открыли на своем компьютере папку с файлами сайта, добавили новые рисунки, схемы или фотографии, отредактировали страницу, подключились к хостингу, закачали новые изображения, заменили страницу, запустили браузер, проверили полученный результат. Если что-то не понравилось, вновь правите страницу и закачиваете ее новую версию на хостинг. Уже чувствуется, что это довольно хлопотное занятие. Между тем, если бы ваш сайт был снабжен CMS, все оказалось бы гораздо проще: запускаете в браузере систему управления и выполняете все манипуляции через нее. Не надо подключаться к хостингу, не надо переносить страницу на свой компьютер, не надо закачивать отредактированную страницу через файловый менеджер. Еще раз повторим: все делается через браузер и без предварительной обработки данных на своем компьютере.

Обычно CMS — это симбиоз двух продуктов: конструктора сайта и системы управления сайтом. В Сети рекламируется много различных CMS, например уже упоминавшиеся во введении WordPress, Joomla, Drupal, 1С-Битрикс и др.

Аббревиатура CMS происходит от английского Content management system, что переводится как система управления содержимым. Многие современные сайты (если не большинство) управляются такими системами. CMS поз-

воляют администратору вносить любые изменения на страницы сайта в онлайн-режиме. Для этого системы управления снабжают WYSIWYG-редакторами.

2.2. WYSIWYG

Сайты пишутся на языках программирования HTML и JavaScript. Поэтому редактирование веб-страницы — это по сути процесс изменения ее кода. Для профессионального разработчика — это привычное занятие. Но ведь большинство сайтов программисты создают не для себя, а для клиентов. Много ли среди них специалистов, досконально знающих HTML и JavaScript? Ответ — единицы. Значит, для большинства клиентов необходим механизм, который бы позволял создавать на сайте новые страницы и редактировать существующие без специальных знаний.

Поэтому креативные программисты придумали не простой редактор, а WYSIWYG (или, как еще говорят, визуальный редактор). Он отображает редактируемые материалы практически в том же самом виде, как и на странице сайта. То есть нажали кнопку «BOLD=» — фрагмент текста действительно выделился жирным. Нажали кнопку «IMG» — и появилась настоящая фотография, а не ее HTML-код.

Термин WYSIWYG — это аббревиатура, произошедшая от английского выражения What You See Is What You Get, что в переводе означает «что вы видите, то вы и получаете». В нашем случае смысл этого выражения таков: что вы видите в редакторе, то и будет на странице сайта. Очень удобно для создания и редактирования контента!

2.3. Атрибут contentEditable

Данный атрибут, если он указан в элементе, сообщает браузеру, что этот элемент — редактируемый.

В чем польза **contentEditable**? Допустим, есть слой **div**, в который мы загружаем предназначенный для редактирования фрагмент веб-страницы. Если у **div** установлен данный атрибут, мы можем вставлять в слой курсор, выделять, удалять, добавлять текст, в том числе вводя его с клавиатуры. То есть у нас уже получился простейший редактор. А если к нему добавить кнопки для импорта в **div** HTML-кода рисунков, таблиц, линий и т. д. то у нас выйдет уже полноценный редактор! Естественно, что написанное с нуля или исправленное содержимое слоя можно будет передать специальной программе для записи в файл или базу данных.

Некоторые программисты указывают атрибут **contentEditable** без значения. Это не совсем корректно, хотя и будет работать. Как утверждает **https://developer.mozilla.org/ru/**, правильно писать:

```
contentEditable="true"
```

Атрибут можно указать непосредственно в теге, например так:

```
<div id="edit" contenteditable="true">
```

или присвоить его в JavaScript-сценарии:

```
document.getElementById("edit").
                setAttribute("contenteditable", true);
```

Обратите внимание! Приведенный выше фрагмент — это пример переноса строки, сделанный в книге из-за недостатка ширины страницы. В реальном файле данный код записывается одной строкой! Автор уже предупреждал о таких переносах во введении. Думаю, читателям все ясно и в дальнейшем не нужно повторять это правило.

Добавлю, что операция присвоения элементу атрибута в JavaScript-сценарии должна производиться после загрузки страницы:

```
addEventListener("load", function()
{
document.getElementById("edit").
           setAttribute("contenteditable", true);
});
```

2.4. Свойство designMode

Свойство **designMode** имеет много общего с **contentEditable**. Отличие в том, что **designMode** устанавливается не для отдельного элемента, а для всего документа в целом. Когда свойство включено (**document.designMode = «on»**), можно редактировать любую HTML-страницу полностью. Эта особенность натолкнула программистов на идею создания визуальных редакторов, используя загрузку целой страницы или ее отдельного фрагмента в **iframe**.

Процедура включения режима редактирования может выглядеть, например, так. Пусть на странице редактора есть фрейм, в который загружается другая HTML-страница, предназначенная для обработки. Тогда в JavaScript-коде редактора необходимо записать всего одну строку:

```
frames[0].document.designMode="on";
```

Естественно, что данная операция, как и в предыдущем случае, должна производиться после загрузки страницы:

```
addEventListener("load", function()
{
frames[0].document.designMode="on";
});
```

Хотя в нашем проекте все манипуляции с контентом производятся непосредственно на редактируемой странице, делается это без использования **designMode**. Мы познакомились с данным свойством в порядке расширения кругозора. В описанной в книге системе принцип действия следующий: выбираете редактируемую область (их на каждой странице 3), нажимаете кнопку и для соответствующего раздела устанавливается атрибут **contentEditable**.

2.5. Метод getSelection

Теперь мы переходим к методам, которые предоставляет в наше распоряжение JavaScript.

Первым упомянем **getSelection**. Тут все очень просто: метод возвращает объект **Selection**, который содержит выделенный пользователем текст. Разработчик имеет возможность на программном уровне манипулировать выделенным фрагментом: форматировать его, копировать, удалять, изменять.

Как выглядит применение метода на практике? Например, у вас есть редактируемая область контента. Вы выделили в этой области одно или несколько слов и нажали кнопку BOLD. К тексту, содержащемуся в объекте **Selection**, будет добавлена пара тегов **b** — открывающий и закрывающий. После этого текущее выделение будет преобразовано к полужирному начертанию.

Аналогично происходит, если вы просто вставили курсор в какое-либо место редактируемой области. Объект **Selection** в этом случае хранит точку вставки курсора. В эту точку можно поместить рисунок, таблицу, линию, символ и т. д.

2.6. Методы createElement и surroundContents

Первый метод позволяет создать новый элемент документа, а второй — добавить его к выделению, полученному методом **getSelection**. Например, у нас в редактируемой области есть слово «Важно», которое мы хотим выделить полужирным шрифтом. Для этого можно написать вот такой сценарий из 4 строк:

```
let sel=document.getSelection();
let rng=sel.getRangeAt(0);
let elm=document.createElement("b");
rng.surroundContents(elm);
```

Разберем, что происходит в каждой строке кода.

1. Создаем объект, содержащий выделенный фрагмент текста.

2. Используя метод **getRangeAt**, создаем объект **Range** с диапазоном, равным выделению. Индекс **0** указывает, что выбранный диапазон у нас первый и единственный.

3. В теле документа методом **createElement** создаем открывающий и закрывающий теги элемента **b** (тег для полужирного начертания шрифта).

4. Методом **surroundContents** добавляем эти теги перед началом выделения и после конца выделения.

Теперь редактируемое слово выглядит так: **Важно**.

Этот способ позволяет добавлять теги не только к выделенному тексту, но и вставлять элементы в положение курсора в редактируемой области: например рисунки, таблицы, линии и т. д.

3. СРЕДА РАЗРАБОТКИ

Для написания и тестирования примеров из книги нам в обязательном порядке необходимо создать на своем компьютере локальный хостинг.

Вообще, хостинг — это, проще говоря, компьютер, предназначенный для размещения сайтов и обеспечивающий к ним доступ из Интернета. Локальный хостинг — набор программ на вашем персональном компьютере, которые позволяют имитировать реальный хостинг и проводить тестирование ваших сайтов (включая HTML-страницы и серверные скрипты) перед их размещением в Сети. **Кстати, обратите внимание: локальный хостинг работает без подключения к Интернету!**

Для чего нужен хостинг в нашем случае? Дело в том, что разбираемая CMS написана не на чистом HTML, а на «смеси» HTML и PHP. То есть каждая страница, которую видит посетитель или администратор системы управления, представляет собой серверную PHP-программу, формирующую данную страницу вслед за запросом. А значит, открыть такую программу просто двойным щелчком по иконке нельзя. Необходимо расположить скрипт на хостинге и запускать, вводя соответствующий запрос в адресном поле браузера. Только в этом случае вы увидите запрошенную страницу.

Чтобы получить результаты, описанные в данной книге, вам понадобится установить на компьютер распространяемый пакет компонентов **Visual C++** от Microsoft, сервер **Apache** и интерпретатор **PHP**. Описание технологии их установки приведено в следующих разделах.

Полноценную среду разработки невозможно представить без настоящего текстового редактора, специально предназначенного для создания программ. Конечно, весь код можно писать в обычном «Блокноте». Но гораздо лучше и рациональнее пользоваться специализированными редакторами. Такие приложения намного удобнее: они подсвечивают код, предлагают синтаксические подсказки, позволяют менять кодировку документов, сохраняют файлы в разных форматах.

В нашем случае необходим редактор, который позволит:

— выполнять качественную разметку;

— создавать файлы с таблицами стилей;

— писать сценарии на JavaScript;

— писать серверные приложения на PHP.

Этим принципам удовлетворяют многие редакторы. Но лучше всего, на наш взгляд, **Notepad++**. Где его взять и как установить, рассказывается в пятом разделе этой главы.

Итак, вводная часть завершена — приступим к созданию локального хостинга. А начнем мы с выяснения разрядности вашей ОС.

Обратите внимание: все процедуры описаны для компьютера с операционной системой Windows 10.

3.1. Выясняем разрядность ОС

Перед тем как мы создадим на компьютере среду разработки, необходимо выяснить разрядность вашей операционной системы, чтобы знать, какие файлы скачивать для локального хостинга.

Включите компьютер, дождитесь полной загрузки операционной системы. Щелкните на кнопке «Пуск», а затем на кнопке «Параметры» (рис. 3.1.1). В открывшемся окне выберите пункт меню «Система». Откроется следующая вкладка, в которой необходимо кликнуть на строке «О системе» (или «О программе»). После этого на вкладке «Характеристики устройства» посмотрите строку «Тип системы» (рис. 3.1.2).

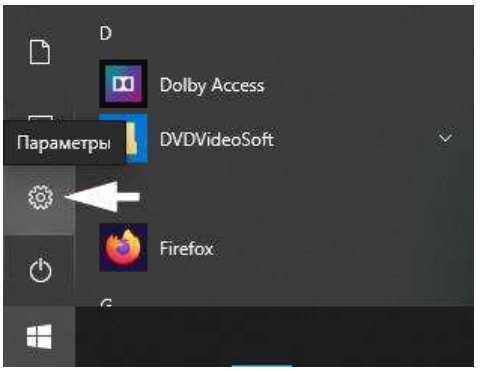

Рис. 3.1.1

Кнопка «Параметры» в меню

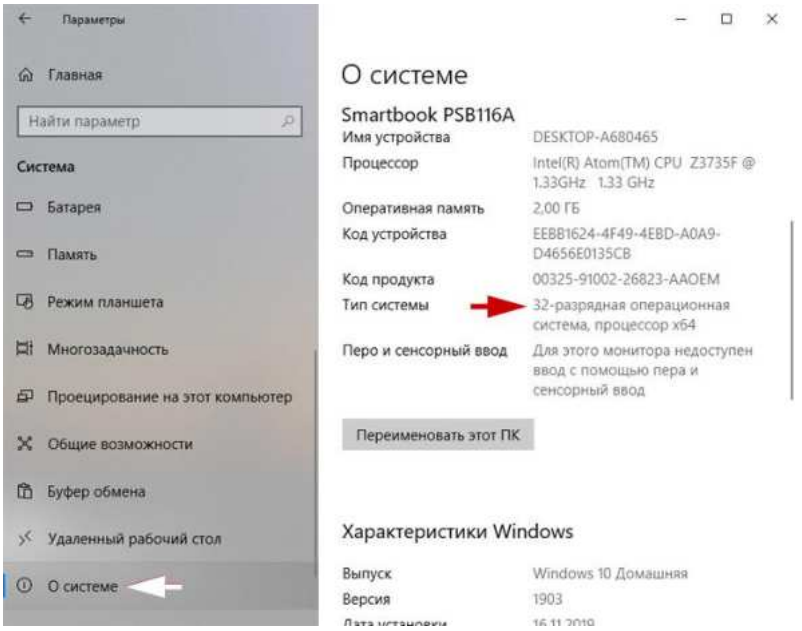

Рис. 3.1.2

Разрядность или тип операционной системы

Типов ОС Windows существует два — 32-разрядная и 64-разрядная. Запомните разрядность вашей. Это число понадобится трижды: при скачивании пакета **Visual C++**, а также zip-архивов сервера Apache и интерпретатора PHP.

3.2. Установка пакета Visual C++

Откройте браузер и зайдите на страницу **https://www.apachelounge.com/-download/**. Это сайт, с которого мы будем скачивать архив с сервером Apache. Но сначала надо установить на ваш компьютер компоненты Visual C++, без которых сервер не заработает.

Для этого найдите на указанной странице текст «**Be sure** you installed latest 14.29.30037.0 Visual C++ Redistributable for Visual Studio 2015–2019: **vc_redist_x64** or **vc_redist_x86** see **Redistributable**» (на момент подготовки книги данные строки были последними в описательной части страницы; со временем текст может несколько измениться). Для нас важны две ссылки, расположенные в тексте: **vc_redist_x64** и **vc_redist_x86**. Если у вас 64-разрядная ОС, нажмите ссылку **vc_redist_x64**. Если 32-разрядная, необходимо щелкнуть на ссылке **vc_redist_x86** (рис. 3.2.1).

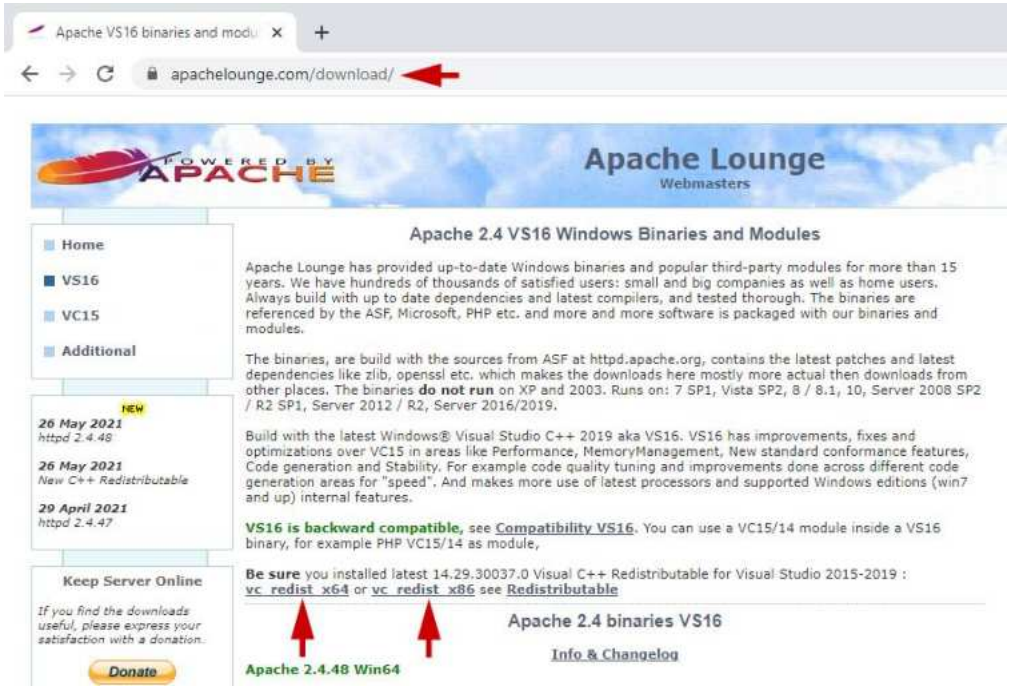

Рис. 3.2.1

Ссылки для скачивания пакетов Visual C++

Запустите скачанный файл (рис. 3.2.2). Примите условия лицензионного соглашения и нажмите кнопку «Установить». Дождитесь завершения процесса установки (рис. 3.2.3). Нажмите кнопку «Закрыть» (рис. 3.2.4).