|  |  |
| --- | --- |
| *voenmeh* | МИНОБРНАУКИ РОССИИ  Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»**  **(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)** |
| БГТУ.СМК-Ф-4.2-К5-01 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Факультет |  | И |  | Информационные и управляющие системы |
|  |  | шифр |  | Наименование |
| Кафедра |  | И9 |  | Систем управления и компьютерных технологий |
|  |  | шифр |  | наименование |
| Дисциплина |  | Научно-исследовательская работа в семестре | | |

ОТЧЕТ

на тему

|  |
| --- |
| Классификация средств разработки сайтов |
|  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил магистрант группы | | | |  | | И9М33 |
| Гаврютина А.А. | | | | | | |
| Фамилия И.О. | | | | | | |
| **ПРЕПОДАВАТЕЛЬ** | | | | | | |
| Каминский В.Н. | |  |  | | | |
| Фамилия И.О. Подпись | | | | | | |
| Оценка |  | | | |  | |
| «\_\_\_\_\_» |  | | | | 2017 г. | |

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc498618158)

[ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc498618159)

[ЭТАПЫ СОЗДАНИЯ ВЕБ-САЙТА 13](#_Toc498618160)

[МЕТОДЫ РАЗРАБОТКИ САЙТОВ 16](#_Toc498618161)

[РУЧНЫЕ МЕТОДЫ РАЗРАБОТКИ САЙТОВ 16](#_Toc498618162)

[АВТОМАТИЗИРОВАННЫЕ СРЕДСТВА СОЗДАНИЯ САЙТОВ 28](#_Toc498618163)

[ЗАКЛЮЧЕНИЕ 35](#_Toc498618164)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 36](#_Toc498618165)

Введение

На сегодняшний день Интернет является неотъемлемой частью жизни современного человека: у людей появился доступ к огромным информационным ресурсам, различным социальным сетям, видео- и аудио-хостингам.

Для создания сайта требуется определить его целевое назначение, то есть что будет происходить на сайте. В зависимости от целевого назначения, предъявляются требования к размещаемому содержимому. Для сайта компании обязательно наличие выходных данных организации, сфера деятельности, сведения о текущих и завершенных проектах, но не обязательно размещение электронных документов, аудио-, видео-файлов, но для сайтов, представляющих собой хранилища файлов последнее обязательно, т.к. это соответствует его целевому назначению.

После определения целевой задачи сайта, пишется техническое задание, в котором оговариваются сроки разработки, исполнители, наименование этапов разработки, а также требования к аппаратным и программным средствам, в том числе и к методам и инструментам разработки.

На сегодняшний день выделяют два основных метода разработки сайтов: методы ручного написания сайтов на одном или нескольких языках веб-программирования и методы автоматизированного создания сайтов - при помощи специальных конструкторов сайтов или же систем управления контентом (СМS).

CMS (от англ. Content Management System) - система управления содержимым (контентом) – это компьютерная программа или информационная система, которая используется для организации и обеспечения процесса по созданию, управлению и редактированию содержимого сайта.

В условиях жесткой конкуренции внутри Web-индустрии одним из важнейших вопросов является скорость разработки сайта.

С увеличением потребности общества в информационных ресурсах, часто для разработки сайтов используются именно методы автоматизированного создания с использованием CMS.

Но такой способ оправдан не всегда, т.к. у сервера, где сайт будет размещен есть ограничения по ресурсам, также может быть ограничение по техническому заданию заказчика на применение инструментов разработки.

Целью исследования является создание классификации средств разработки сайтов по критерию целевой задачи.

Задачами исследования являются:

* Произвести анализ предметной области.
* Рассмотреть принципы работы архитектуры клиент-сервер.
* Рассмотреть принципы работы CMS.
* Рассмотреть основные типы сайтов и целевых задач сайта.

Объектом исследования является совокупность средств разработки сайтов. Предметом исследования является свойства, по которым производится классификация средств разработки сайтов (в данном случае – целевая задача сайта).

Прикладная ценность и новизна исследования обусловлена тем, что на данный момент не существует классификации средств разработки сайтов по типу задачи, который будущий сайт будет выполнять.

Описание предметной области

Для описания предметной области введем определение, что такое веб-программирование.

Веб-программирование – это область программирования, служащая для разработки сайтов.

Сайт – это совокупность веб-страниц, объединённых общей темой и хранящихся на одном веб-сервере.

Веб-страница – это документ, хранящийся на веб-сервере, или создаваемый по запросу пользователя и имеющий уникальный адрес. Во всемирной паутине адрес называют URL (от англ. Uniform Resource Locator ­­– унифицированный указатель ресурсов). URL главной страницы сайта называют адресом сайта. Веб-страница имеет следующие синонимы:

* страница;
* HTML-страница;
* гипертекстовый документ;
* HTML-документ;
* веб-документ.

Веб-страницы делятся на:

* статические и динамические;
* активные и пассивные.

Статическая страница хранится в готовом виде на веб-сервере и выдаётся по запросу пользователя. В самом простом случае статическая страница состоит из одного файла, содержащего текст, размеченный тегами языка HTML. В общем случае в состав страницы входят файлы с изображениями, аудио- и видеофайлы, файлы с каскадными таблицами стилей (CSS) и файлы, содержащие скрипты, выполняемые на стороне браузера. HTML-файл содержит ссылки на все остальные файлы страницы [1].

Скриптом в веб-программировании называют программу, служащую для создания или изменения страницы. Различают скрипты, выполняемые на стороне веб-сервера и на стороне браузера. Серверные скрипты обычно пишут на языках Perl, PHP, Node.js, а для браузерных используют JavaScript.

Динамическая страница создаётся (генерируется) на веб-сервере скриптом по запросу пользователя. В остальном, она ничем не отличается от статической.

Свойства страницы, поступившей с веб-сервера на браузер, не зависят от того, динамическая она или статическая. Страница, вид которой в окне браузера в течение всего времени просмотра остаётся неизменным, называется пассивной. Страница, меняющая свой вид под воздействием пользователя (движением или щелчком мышки, нажатием клавиши) называется активной.

Активная страница содержит в себе скрипт, выполняемый на стороне браузера в ответ на действия пользователя.

Итак, страница одновременно может быть:

* динамической и активной,
* динамической и пассивной,
* статической и активной,
* статической и пассивной.

Разделение страниц на статические и динамические позволяет существенно повысить производительность веб-серверов. Время выполнения запроса на выдачу статической страницы в среднем намного меньше, чем на выдачу динамической, так как для формирования последней требуется выполнить скрипт. Значительная часть времени выполнения скрипта приходится на запросы к базе данных. Веб-сервер, способный организовать создание динамических страниц, имеет сложную структуру и поэтому работает относительно медленно. Поэтому созданы более простые и быстрые лёгкие веб-серверы, предназначенные для обработки только статических страниц [2].

Термин веб-сервер имеет 3 различных значения:

* программа, посылающая по запросу пользователя гипертекстовый документ (такую программу будем называть программным веб-сервером),
* компьютер, на котором установлен программный веб-сервер,
* вычислительная система, состоящая из множества компьютеров с установленными на них программными веб-серверами (такую вычислительную систему будем называть веб-кластером).

Статической принято называть страницу, которая целиком хранится на сервере и показывается посетителю в своем неизменном виде (следует учесть, что статическая страница может содержать некоторые изменяемые элементы, например, баннеры, однако она все равно остается статической). Для большей ясности можно пояснить понятие следующим образом. Вы, вероятно, знаете, что в каждом браузере имеется возможность просмотреть так называемый исходный код страницы. Так вот, если на сервере страница хранится в точности в таком же виде, то это статическая страница. Сервер просто пересылает такую страницу браузеру, который выводит ее на экран монитора (этот процесс называют "рендерингом") и тем самым выдает их посетителю вашего сайта [3].

Статическим называется сайт, все страницы которого являются статическими. Таких сайтов довольно много, с них, можно сказать, начинался интернет. Статические сайты создаются, в основном, на "чистом HTML" (ну, допустимо использование CSS).

* Преимущества систем на базе языка HTML:
* Легко изменить внешний вид любой конкретной страницы, не повлияв на вид других страниц.
* Несложно добавить на сайт новую страницу, скопировав и исправив файл с существующей страницей.
* Человеку, занимающемуся поддержкой сайта, достаточно знать только язык HTML.
* Сайт будет работать на любом сервере хостинга, даже с самыми ограниченными возможностями.

Сайт можно просматривать локально, не устанавливая дополнительное программное обеспечение. Веб-мастеру удобно отлаживать страницы, не выходя в Интернет.

Малое число используемых программных компонентов делает затруднительным взлом такой системы.

Однако, при всех своих преимуществах такие системы обладают значительными недостатками:

Сложно внести изменения в структуру и внешний вид сайта. Например, если надо изменить меню сайта, логотип или структуру страниц, то для этого необходимо изменить содержимое всех страниц, поскольку HTML-код этих элементов продублирован на каждой странице.

Система не гарантирует единый стиль страниц сайта - каждая страница фактически создается заново и любая ошибка при изменении файлов, содержащих страницы, приводит к тому, что разные страницы сайта выглядят по-разному.

Невозможно использовать такие компоненты, как комментарии и отзывы пользователей, голосования, форум, чат и т.д.

Невозможно или достаточно трудно обеспечить разделение прав доступа к содержимому сайта.

За годы использования таких систем были разработаны решения, частично компенсирующие перечисленные выше недостатки статических сайтов. Первый недостаток может быть частично устранен путем вынесения меню и общих элементов навигации в отдельные файлы, что может быть сделано с помощью технологии серверных включений SSI - [Server Side Includes](http://website-ru.net/txt/basic/ssi.shtml) или путем использования фреймов. Второй недостаток устраняется тем же путем за счет заполнения шаблона страницы какой-либо информацией, хранящейся отдельно или получаемой в результате работы алгоритма (например, в результате вычислений). Третий недостаток может быть исключен с помощью добавления отдельных программ, написанных на языке PHP. Четвертый недостаток может быть частично компенсирован путем размещения страниц в различные папки и назначением различных прав доступа к этим папкам. Естественно, что самым распространенным является случай, когда названные и другие варианты формирования странички используются в различных сочетаниях.

Системы на базе HTML с использованием дополнительных решений до сих пор широко используются. Однако поскольку в эти системы добавляются в том или ином виде элементы программирования, постепенно происходит переход к современным "динамическим" системам управления сайтом.

Динамической будем называть страницу, формируемую сервером из нескольких частей или получаемую путем внесения либо изменения каких-то данных в хранящуюся на сервере заготовку страницы. То есть динамическая страница, в отличие от статической, собирается каким-то способом из данных, хранящихся на сервере, и только после этого показывается посетителю.

Динамическим следует называть любой сайт, на котором имеется хотя бы одна динамическая страница. Дело в том, что для создания даже одной динамической страницы нужно уже задействовать те механизмы, которые позволяют такие страницы создавать. И если владелец сайта такие механизмы освоил, число динамических страниц на его сайте будет обязательно увеличиваться, а доля статических страниц, соответственно, уменьшаться.

Дальнейшим развитием идеи динамического сайта являются интерактивные сайт — это сайты, который активно взаимодействуют с пользователем, выдавая информацию, различающуюся в зависимости от действий пользователя. Самый характерный пример - игры с удаленным компьютером через Интернет. Но есть и более практические применения интерактивности: поиск, обращение к базам данных, различные регистрации, формы и прочее. Интерактивность на персональной странице позволяет привлечь больше посетителей, поскольку повышает удобство пользования информацией на сайте (примеры: смена кодировки, языка, выбор дизайна и так далее).

Очевидно, что чтобы сделать сайт динамическим, то есть чтобы у вебмастера появилась возможность создавать динамические страницы, на стороне сервера должны работать какие-то программы, которые формируют выдаваемую страничку из различных частей (создаваемых, возможно, другими программами) и обрабатывают данные, присылаемые пользователями. Следовательно, вебмастер должен при создании такого сайта использовать средства программирования. К ним относятся CGI, PHP, Perl, ASP и т.д.

Преимущества динамических сайтов:

1. Главное достоинство динамических сайтов — это возможность быстрого внесения изменений сразу во все страницы сайта.
2. Упрощение модификации и обновления страниц на сайте.
3. Разделение информации и дизайна сайта. Дизайн сайта - понятие многогранное. Можно понимать дизайн в широком и в узком смысле. Под дизайном в узком смысле понимают только внешний вид сайта: набор цветов или фоновых картинок, выбор шрифтов для текста и заголовков, шаблон страницы (выделение определенных областей страницы для тех или иных блоков, образующих страницу). Использование динамических страниц позволяет именно дизайн в этом узком смысле отделить от содержательного наполнения страницы (контента).
4. Возможность для посетителей изменять контент сайта. Преимущества, описанные в первых двух пунктах, облегчали работу создателей сайтов и снижали их трудозатраты, но для посетителей это было не очень важно. А вот возможность изменять содержание страниц отразилась непосредственно на них. Но очевидно, что если на сайте планируется какая-либо обратная связь с пользователем, например, предоставление пользователю возможности добавлять комментарии по поводу той или иной статьи, или голосование, то никак нельзя обойтись статическими документами.
5. Таким образом, видно, что обойтись одними только статическими документами при создании современного веб-сайта в подавляющем большинстве случаев невозможно. Большая часть страниц, пусть даже с неизменным содержимым, все равно должна собираться динамически, из-за наличия меняющихся меню, элементов оформления и т.д. Такой сайт скорее представляет из себя не набор документов, а программу, работающую на сервере.

Недостатки динамических сайтов

1. Появляется необходимость использования дополнительных программных средств для построения динамического сайта. На статическом сайте все страницы уже готовы, серверу остается только показать их посетителю, а на динамическом сайте необходимо вносить в них какие-то изменения, для этого требуется соответствующие программные решения.
2. В зависимости от сложности сайта, трудоемкость и стоимость разработки таких программ может очень сильно варьироваться. Сейчас существует множество готовых решений для создания сайта, в том числе и бесплатных. В сети можно найти не один десяток всевозможных бесплатных скриптов, которые позволят вам создать на сайте форум, доску объявлений, клуб знакомств, магазин и т.д. Но, если требуется что-то специфическое, все равно не обойтись без дополнительных разработок.
3. Повышение требований к аппаратным мощностям серверных систем. Этот недостаток непосредственно следует из предыдущего, т.к. теперь серверу требуется еще выполнить какую-то программу для модификации страницы сайта, а только потом выдать ее посетителю. Особенно заметной эта проблема становится на сайтах с большой посещаемостью. Часто в таких случаях приходится производить дополнительную оптимизацию кода для нормальной работы сайта.
4. Следовательно, и стоимость услуг по содержанию такого сайта, намного выше, нежели статического. Хотя сейчас даже многие бесплатные хостинги поддерживают возможность создания динамических сайтов, не говоря уже о платных, где все необходимое входит в стандартный набор услуг.
5. Третьим недостатком, также вытекающим из первого, является сложность больших структурных изменений сайта. Вы можете возмутиться, ведь выше было сказано, что динамика облегчает и упрощает модификацию. Но не спешите, давайте разберем этот вопрос подробнее. В данном случае все завязано на программное обеспечение, которое используется на сайте. Именно оно объединяет кусочки дизайна и данных в один полноценный сайт. Но у любой программы есть свои ограничения, и, если требуется получить что-то, что программно не предусмотрено, значит нужно менять программу. Либо дорабатывать старую, либо искать новую [3].

Несмотря на эти недостатки, динамических сайтов в сети становится все больше и больше, видимо, перечисленные выше преимущества перекрывают все недостатки.

Надо сказать, что сейчас все больше набирает силу гибридная схема разработки. Речь идет об AJAX, что означает: Asyncronous JavaScript And XML. Эта технология позволяет скриптам на JavaScript обращаться к какому-либо скрипту на сервере и получать информацию с сервера. Это в свою очередь, дает пользователю гибкость и позволяет перезагружать только часть содержимого страницы, а не всю ее полностью (что значительно экономит трафик).

Этапы создания веб-сайта

1 этап — определение целей создания сайта. Это самый важный этап в создании сайта, потому что нельзя добиться цели, если её нет или если она определена неправильно. От целей будет зависеть весь дальнейший процесс создания сайта, каждый его этап. Поэтому к постановке цели нужно подходить с максимальной серьёзностью.

2 этап — проведение исследований по теме

Прежде, чем преступить к разработке, необходимо проанализировать тему, изучить сайты потенциальных конкурентов. В дальнейшем это поможет в создании собственной концепции.

3 этап — определение типа сайта, разработка ТЗ и структуры

На этом этапе разработчик совместно с заказчиком составляет ТЗ (техническое задание) на создание сайта. ТЗ, как минимум, должно включать следующие пункты:

* Тип сайта (портал, промо, визитка, корпоративный, магазин и т.д.)
* Функциональные возможности сайта (поиск, каталог, лента новостей и т.д.)
* Стиль дизайна (строгий, веселый, мрачный и т.д.)
* Структура сайта (какие страницы должны быть на сайте)
* Структура страниц (какие блоки должны быть на страницах и в каком виде)

При составлении ТЗ обязательно нужно помнить о целях создания сайта. Если какая-нибудь функция сайта не способствует достижению поставленных целей (или тем более — мешает) — необходимо от неё отказаться: на сайте она будет лишней. Например, если цель сайта — продать как можно больше единиц товара, то лента новостей на сайте не нужна.

4 этап — разработка макета дизайна сайта

Этот этап делится на несколько подэтапов:

* Генерация идей дизайна. Набор идей предоставляется заказчику в виде эскизов с текстовыми пояснениями.
* Разработка предварительного макета дизайна главной страницы. Исправление замечаний заказчика (если есть)
* Доработка макета до завершенного вида.
* Разработка внутренних страниц по аналогичному алгоритму. В макете должны быть прорисованы все блоки, которые будут на сайте.

Если к моменту разработки дизайна текстовые и графические материалы ещё не готовы, можно использовать «рыбу», но нельзя оставлять в макете «пустые» места.

Естественно, каждый шаг разработки макета дизайна должен выполняться в соответствии с ТЗ и целями сайта.

5 этап — HTML-CSS вёрстка

В зависимости от целей и задач сайта, верстка должна удовлетворять некоторым требованиям. Обычно эти требования такие:

* Кроссбраузерность — страницы должны одинаково отображаться в разных браузерах одинаково корректно (Mozilla Firefox, Google Chrome, Opera, Internet Explorer, Safari и т.д.).
* Гибкость — возможность легко добавлять/удалять информацию на страницы.
* Быстрота обработки кода браузером
* Валидность — соответствие стандартам
* Семантическая корректность — логичное и правильное использование элементов HTML

6 этап — программирование и установка на CMS

Это чисто технический этап, на котором реализуется весь набор функций, требуемых для сайта. Требования к этому этапу определяются техническим заданием.

7 этап — заполнение сайта контентом (информацией)

На этом этапе очень важен качественный, профессиональный копирайтинг. Все материалы сайта, будь то тексты или графика, также должны вписываться в общую концепцию сайта, соответствовать его целям и задачам.

8 этап — тестирование сайта и исправление ошибок

Тестирование сайта может проводить как разработчик, так и заказчик. Наилучший вариант — это совместное тестирование.

9 этап — публикация сайта в Интернет

На этом этапе сайт размещается на выбранном доменном имени, регистрируется в крупных поисковиках и каталогах.

10 этап — продвижение сайта и реклама в интернете

Когда сайт полностью готов к работе — нужно привлекать на него посетителей. Для продвижения сайта можно воспользоваться контекстной или баннерной рекламой, SEO, SMO и другими методами.

Методы разработки сайтов

Ручные методы разработки сайтов

Условно, методы разработки сайтов можно разделить на две группы – ручные и автоматизированные методы.

Ручной метод разработки сайта предполагает написание программистом самостоятельно страниц сайта и организация взаимодействия между ними и сервером.

Для создания веб-страниц используются следующие средства:

1. Язык разметки HTML

HTML (от англ. HyperText Markup Language) представляет собой довольно простой набор команд, описывающих структуру документа. Этот язык разметки позволяет выделить в документе отдельные логические части – заголовки, абзацы, таблицы, списки-перечисления и т.д., но не задает конкретные атрибуты форматирования. Конкретный вид форматирования определяет сам браузер при чтении документа, и именно браузер обеспечивает наилучшее отображение Web-документа на вашем экране.

Создавать Web-страницы можно с помощью специальных программ-редакторов, автоматически генерирующих код HTML, работа с которыми не требует знания языка разметки. Однако эти программы часто ограничены в своих возможностях, содержат ошибки и нередко создают плохой HTML-код, который работает не на всех платформах.

Язык HTML существует в нескольких вариантах или спецификациях. Как и версии программных продуктов, спецификации пронумерованы: 2.0, 3.0, 3.2, 4.0. Каждая последующая спецификация представляет собой расширение и дополнение предыдущей таблицы, списки-перечисления и т.д., но не задает конкретные атрибуты форматирования. Конкретный вид форматирования определяет сам браузер при чтении документа, и именно браузер обеспечивает наилучшее отображение Web-документа на вашем экране [2].

Документ в окне с кодом HTML — это текстовый документ специального формата. Все файлы этого формата имеют расширение .html или .htm. В документе HTML обычный текст сочетается с элементами разметки, заключенными в угловые скобки < и >, например< html>, <head>, <title>, </title>. Такие элементы разметки называются тэгами. Тэги бывают одиночными, открывающими и закрывающими и состоят из следующих друг за другом в определенном порядке элементов:

* + левой угловой скобки< ;
  + необязательного символа слэш /, который означает, что тэг является конечным тэгом, закрывающим некоторую структуру, например, </title>. В этом смысле можно читать символ / как конец некоторого элемента разметки, например, строки или абзаца;
  + имени тэга, например, html;
  + необязательных атрибутов. Тэг может быть без атрибутов или сопровождаться одним, или несколькими атрибутами, например, align="center";
  + правой угловой скобки> .

В тэгах могут использоваться только символы латинского алфавита, а в значениях атрибутов - любые символы. Если в качестве значений атрибута используются, например, символы кириллицы, то они должны быть заключены в кавычки, например, name="Раздел 1".

Язык HTML не различает большие и малые буквы, так что тэги <HEAD>, <head> и <Head> эквивалентны.

Большинство тэгов парные: за открывающим тэгом следует соответствующий ему закрывающий тэг, а между ними содержится текст или другие тэги [2].

В таких случаях два тэга и часть документа, заключенная между ними, образует блок, называемый элементом HTML. Некоторые тэги, например, <br>, являются одиночными, и для них закрывающий тэг не применяется. Такие тэги сами по себе являются элементами HTML.

Большинство тэгов могут иметь один или несколько атрибутов - параметров, дающих дополнительную информацию о том, как браузер должен обрабатывать текущий тэг. Однако атрибутов может и не быть вовсе. Атрибут тэга состоит из имени, например, align, знака равенства = и значения, которое задается строкой символов, например, "center":align="center". Значения атрибутов обычно заключаются в кавычки.

Каждый HTML-документ имеет определенную структуру, которая выглядит следующим образом:

<html>

<head>

…

</head>

<body>

…

</body>

</html>

Структура HTML-документа содержит следующие обязательные элементы:

* тэги <html> и </html>, которые отмечают начало и конец документа;
* заголовок, ограниченный тэгами <head> и </head>;
* тело, ограниченное тэгами <body>...</body>.

В заголовке, ограниченном тэгами <head> и </head>, с помощью тэгов <title>...</title> определяется название документа, которое должно описывать его содержимое и обычно содержит не более 5-6 слов. Это название отображается браузерами в строке заголовка рабочего окна программы, а роботы, составляющие индексы для поисковых систем, идентифицируют документ, используя его название.

Кроме элемента <title>...</title>, заголовок может содержать элементы < meta>...</meta>. Открывающий тэг <meta> включает пары имя=значение, описывающие свойства документа, например, авторство, список ключевых слов и т.д. Эти данные используются также поисковыми серверами при индексации документов.

1. Язык JavaScript

При генерации страниц в Web возникает дилемма, связанная с архитектурой «клиент-сервер». Страницы можно генерировать как на стороне клиента, так и на стороне сервера. В 1995 году специалисты компании Netscape создали механизм управления страницами на клиентской стороне, разработав язык программирования JavaScript.

Когда создавался язык JavaScript, у него изначально было другое название: «LiveScript». Но тогда был очень популярен язык Java, и маркетологи решили, что схожее название сделает новый язык более популярным.

JavaScript - это язык управления сценариями просмотра гипертекстовых страниц на стороне клиента. Если быть более точным, то JavaScript - это не только язык программирования на стороне клиента. Liveware, прародитель JavaScript, является средством подстановок на стороне сервера Netscape. Однако наибольшую популярность JavaScriptобеспечило программирование на стороне клиента.

Основная идея JavaScript состоит в возможности изменения значений атрибутов HTML-контейнеров и свойств среды отображения в процессе просмотра HTML-страницы пользователем. При этом перезагрузки страницы не происходит.

На практике это выражается в том, что можно, например, изменить цвет фона страницы или интегрированную в документ картинку, открыть новое окно или выдать предупреждение.

JavaScript стандартизован ECMA (англ. European Computer Manufacturers Association, Ассоциация европейских производителей компьютеров). Соответствующие стандарты носят названия ECMA-262 и ISO-16262. Этими стандартами определяется язык ECMAScript, который примерно эквивалентен JavaScript 1.1. Отметим, что не все реализации JavaScript на сегодня полностью соответствуют стандарту ECMA [4].

Современный JavaScript – это «безопасный» язык программирования общего назначения. Он не предоставляет низкоуровневых средств работы с памятью, процессором, так как изначально был ориентирован на браузеры, в которых это не требуется.

Что же касается остальных возможностей, они зависят от окружения, в котором запущен JavaScript. В браузере JavaScript умеет делать всё, что относится к манипуляциям со страницей, взаимодействиям с посетителем и, в какой-то мере, с сервером:

* Создавать новые HTML-теги, удалять существующие, менять стили элементов, прятать, показывать элементы и т.п.
* Реагировать на действия посетителя, обрабатывать клики мыши, перемещения курсора, нажатия на клавиатуру и т.п.
* Посылать запросы на сервер и загружать данные без перезагрузки страницы (эта технология называется AJAX).
* Получать и устанавливать cookie, запрашивать данные, выводить сообщения и т.д. [5]

Большинство возможностей JavaScript в браузере ограничено текущим окном и страницей.

JavaScript не имеет прямого доступа к операционной системе.

Современные браузеры могут работать с файлами, но эта возможность ограничена специально выделенной директорией – «песочницей». Возможности по доступу к устройствам также прорабатываются в современных стандартах и частично доступны в некоторых браузерах.

JavaScript, работающий в одной вкладке, не может общаться с другими вкладками и окнами, за исключением случая, когда он сам открыл это окно или несколько вкладок из одного источника (одинаковый домен, порт, протокол).

Из JavaScript можно легко посылать запросы на сервер, с которого пришла страница. Запрос на другой домен тоже возможен, но менее удобен, т. к. и здесь есть ограничения безопасности.

Для создания механизма управления страницами на клиентской стороне было предложено использовать объектную модель документа (англ. Document Object Model, DOM). Суть модели в том, что каждый HTML-контейнер - это объект, который характеризуется тройкой:

* свойства;
* методы;
* события.

Объектную модель можно представить как способ связи между страницами и браузером. Объектная модель - это представление объектов, методов, свойств и событий, которые присутствуют и происходят в программном обеспечении браузера, в виде, удобном для работы с ними кода HTML и исходного текста сценария на странице.

Объекты с одинаковым набором свойств, методов и событий объединяются в классы однотипных объектов. Классы - это описания возможных объектов. Сами объекты появляются только после загрузки документа браузером или как результат работы программы. Об этом нужно всегда помнить, чтобы не обратиться к объекту, которого нет.

1. Cвойства

Многие HTML-контейнеры имеют атрибуты. Например, контейнер якоря <A...>...</A> имеет атрибут HREF, который превращает его в гипертекстовую ссылку:

<A HREF=example.htm> example </A>

Если рассматривать контейнер якоря <A ...>...</A> как объект, то атрибут HREF будет задавать свойство объекта «якорь». Программист может изменить значение атрибута и, следовательно, свойство объекта:

document.links[0].href=" example.htm";

Не у всех атрибутов можно изменять значения. Например, высота и ширина графической картинки определяются по первой загруженной в момент отображения страницы картинке. Все последующие картинки, которые заменяют начальную, масштабируются до нее [5].

Для общности картины свойствами в JavaScript наделены объекты, которые не имеют аналогов в HTML-разметке. Например, среда исполнения, называемая объектом Navigator, или окно браузера, которое является вообще самым старшим объектом JavaScript.

1. Методы

В терминологии JavaScript методы объекта определяют функции изменения его свойств. Например, с объектом «документ» связаны методы *open(),* *write(), close().* Эти методы позволяют сгенерировать или изменить содержание документа. Приведем простой пример:

function test()

{

id=window.open("","example","width=400, height=150");

id.focus(); id.document.open();

id.document.write("<H2>Привет!</H2>");

id.document.write("<HR><FORM>");

id.document.write("<INPUT TYPE=button VALUE='Закрыть окно' ");

id.document.write("onClick='window.opener.focus();

window.close();'>");

id.document.close();

}

В данном примере метод *open()* открывает поток записи в документ, метод *write()* осуществляет эту запись, метод *close()* закрывает поток записи в документ. Все происходит так же, как и при записи в обычный файл. Если у окна есть поле статуса, то при незакрытом потоке записи в документ в нем будет «метаться» прямоугольник продолжения записи, как это происходит при загрузке документа [5].

1. События

Кроме методов и свойств, объекты характеризуются событиями. Например, с объектом типа *button* (контейнер *INPUT* типа *button* – «Кнопка») может происходить событие *click*, т.е. пользователь может нажать на кнопку. Для этого атрибуты контейнера *INPUT* расширены атрибутом обработки события *click* - *onClick*. В качестве значения этого атрибута указывается программа обработки события, которую должен написать на JavaScript автор HTML-документа:

<INPUT TYPE=button VALUE="Нажать" onClick=

"window.alert('Пожалуйста, нажмите еще раз');">

Обработчики событий указываются в тех контейнерах, с которыми эти события связаны. Например, контейнер *BODY* определяет свойства всего документа, поэтому обработчик события завершения загрузки всего документа указывается в этом контейнере как значение атрибута *onLoad****.***

Строго говоря, каждый браузер, Internet Explorer, Google Chrome или Opera, имеет свою объектную модель. Объектные модели разных браузеров (и даже разные версии одного) отличаются друг от друга, но имеют принципиально одинаковую структуру.

Серверная часть

1. PHP

PHP – это язык программирования, используемый на стороне web-сервера для динамической генерации HTML-страниц. PHP расшифровывается как Personal HyperText Processor. PHP является одиним из немногих языков программирования, созданных специально для разработки веб-приложений.

Поэтому он включает в себя все функции, необходимые именно для работы на веб-сервере, и при этом лишен избыточности.  
Особенность PHP заключается в том, что его команды включаются в обычные HTML-страницы с помощью специальных тегов. Программам на PHP не нужны специальные CGI-директории с особыми правами доступа. Более того, на одной страничке можно произвольно чередовать HTML и PHP-код.  
PHP интегрируется во все популярные веб-серверы: Apache и IIS, Zens и др., работает под Windows и с практически всеми UNIX-подобными системами. Как следствие, PHP работает практически у всех хостинг-провайдеров, разрешающих собственные выполняемые скрипты [6].

PHP поддерживает большинство современных веб-протоколов: IMAP, FTP, POP, SMTP и другие. PHP также умеет работать с базами данных.

Современные тенденции развития языков программирования не обошли стороной и PHP. Средства объектно-ориентированного программирования появились еще в PHP3. А в объектной модели PHP4 в полном объеме реализованы классические понятия объектно-ориентированного программирования: наследование, инкапсуляция и полиморфизм [7].

1. Perl

Perl — это высокоуровневый интерпретируемый динамический язык программирования общего назначения, созданный Ларри Уоллом, лингвистом по образованию. Название языка официально расшифровывается как «практический язык для извлечения данных и составления отчётов» (англ. Practical Extraction and Report Language), а в шутку как «патологически эклектичный перечислитель мусора» (англ. Pathologically Eclectic Rubbish Lister). Символом языка Perl является верблюд — не слишком красивое, но очень выносливое животное, способное выполнять тяжёлую работу [8].

Основной особенностью языка считаются его богатые возможности для работы с текстом, в том числе работа с регулярными выражениями, встроенная в синтаксис. Перл унаследовал много свойств от языков Си, AWK, скриптовых языков командных оболочек UNIX.

Perl также знаменит огромной коллекцией дополнительных модулей CPAN, находящейся по адресу http://www.cpan.org.

Perl — язык программирования общего назначения, который был первоначально создан для манипуляций с текстом, но на данный момент используется для выполнения широкого спектра задач, включая системное администрирование, веб-разработку и т.д.

Главными достоинствами языка являются поддержка различных парадигм (процедурный, объектно-ориентированный и функциональный стили программирования), контроль за памятью (без сборщика мусора, основанного на циклах), встроенная поддержка обработки текста, а также большая коллекция модулей сторонних разработчиков. Все версии Perl выполняют автоматическую типизацию данных и автоматический контроль над памятью. Интерпретатор знает тип и запросы памяти каждого объекта программы, он распределяет и освобождает память, производя подсчёт ссылок.

Perl также заимствует ряд свойств из языков программирования командных оболочек UNIX. Все переменные маркируются ведущими знаками, которые точно выражают тип данных переменной в этом контексте (например, скаляр, массив, хеш). Эти знаки позволяют переменным быть интерполированным в строках. Perl обладает множеством встроенных функций, которые обеспечивают инструментарий, часто используемый для программирования оболочки, например, сортировку или вызов системных служб.

Первая строка исходного кода может начинаться с «#!/Путь/к/Perl [-ключи]», что указывает системе путь к интерпретатору Perl для выполнения программы в UNIX-системах и выполнения их на веб-сервере [9].

Простейшая программа «Hello, world!» выглядит следующим образом:

#!/usr/bin/perl

print "Hello, world!\n";

либо для CGI:

#!/usr/bin/perl

print "Content-type: text/html\n\n";

print "Hello, world!";

1. NODE JS

Node (другое название – Node.js) – это появившаяся платформа, которая выводит язык JavaScript за пределы браузера и позволяет использовать его в серверных приложениях. В основе платформы лежит исключительно быстрый движок JavaScript, заимствованный из браузера Chrome, V8, к которому добавлена быстрая и надежная библиотека асинхронного сетевого ввода/вывода. Основной упор в Node делается на создании высокопроизводительных, хорошо масштабируемых клиентских и серверных приложений для «веб реального времени».

Эту платформу разработал Райан Дал в 2009 году, после двух лет экспериментирования с созданием серверных веб-компонентов на Ruby и других языках. В ходе своих исследований он пришел к выводу, что вместо традиционной модели параллелизма на основе потоков следует обратиться к событийно-ориентированным системам. Эта модель была выбрана за простоту (хорошо известно, что многопоточные системы трудно реализовать правильно), за низкие накладные расходы, по сравнению с идеологией «один поток на каждое соединение», и за быстродействие. Цель Node - предложить «простой способ построения масштабируемых сетевых серверов». При проектировании за образец были взяты такие системы, как EventMachine (Ruby) и каркас Twisted (Python).

Node - платформа для написания JavaScript-приложений вне веб-браузера. В Node не встроена ни объектная модель документа (DOM), ни какие-либо ещё возможности браузера. Именно язык JavaScript в сочетании с асинхронным вводом/выводом делает Node мощной платформой для разработки приложений. Но вот для чего Node непригодна, так это для разработки персональных приложений с графическим интерфейсом пользователя (ГИП). На сегодняшний день в Node нет встроенного эквивалента Swing (или SWT). Нет и подключаемой библиотеки ГИП для Node, и внедрить Node в браузер тоже нельзя. Если бы для Node существовала библиотека ГИП, то на этой платформе можно было строить и персональные приложения. Недавно появилось несколько проектов по созданию интерфейса между Node и GTK, итогом которых должна стать кросс-платформенная библиотека ГИП [10].

Помимо встроенного умения исполнять код на JavaScript, включенные в состав дистрибутива модули предоставляют и другие возможности:

* утилиты командной строки (для включения в скрипты оболочки);
* средства написания интерактивных консольных программ (цикл «чтение - выполнение - печать»);
* великолепные функции управления процессами для наблюдения за дочерними процессами;
* объект Buffer для работы с двоичными данными;
* механизм для работы с сокетами TCP и UDP с полным комплектом обратных вызовов в ответ на события;
* поиск в системе DNS;
* средства для создания серверов и клиентов протоколов HTTP и HTTPS, построенные на основе библиотеки ТСР-сокетов;
* средства доступа к файловой системе;
* встроенная рудиментарная поддержка автономного тестирования с помощью утверждений.

Автоматизированные средства создания сайтов

Помимо «ручного» способа создания веб-сайтов, также следует рассмотреть автоматизированные средства разработки сайтов.

Под автоматизированным способом разработки будет считать способ разработки веб-сайта с использованием автоматизированных средств. К таким можно отнести системы управления содержимым (англ. Content Management System, CMS) и фреймворк для управления содержимым (англ. Content Management Framework, CMF).

Рассмотрим каждое из средств подробнее.

1. Системы управления содержимым

CMS (от англ. Content Management System) – это система управления содержимым (контентом) – компьютерная программа или информационная система, которая используется для организации и обеспечения процесса по совместному созданию, управлению и редактированию содержимого сайта.

Основной задачей такой системы является сбор и объединение в единое целое, на основе ролей и задач, различных источников информации. Эти источники могут быть доступны как внутри самой организации, так и вне её пределов. К тому же данная система обеспечивает возможность взаимодействия различных сотрудников, проектов и рабочих групп, с теми базами знаний и данных, которые были ранее созданы, в таком виде и таким способом, чтобы сделать процесс поиска и повторного использования максимально комфортным и привычным [11].

В такой системе управления контентом определяется всё многообразие существующих данных: стандартные документы, музыка и звуки, видео, каталоги всевозможной информации и многое-многое другое. И именно для управления, хранения, обработки, просмотра и публикации таких данных различными группами пользователей и служат CMS. Отсюда кстати возникает и новый род профессиональной деятельности - контент менеджер, или проще говоря - редактор сайта

Разработка сайта на основе какой-либо CMS имеет следующие преимущества:

1. использование CMS позволяет владельцу сайта самостоятельно создавать и удалять разделы сайта, редактировать различную информацию без привлечения стороннего специалиста;
2. низкий порог вхождения разработчика;
3. временные затраты на разработку сайта существенно снижаются, так как разработчику не надо фиксировать своё внимание на чисто технических задачах, а можно сосредоточиться на информационной и визуальной составляющих будущего сайта.

Некоторые CMS ориентированы только на решение конкретных задач (ведение блогов, интернет-магазины, форумы), другие являются универсальными и предоставляют разработчиком удобную среду проектирования и программирования для разработки чего угодно.

Часть CMS состоять из множества функциональных блоков и модулей, другие монолитны, неделимы. Одни системы поставляются бесплатно и с возможностью внесения своих доработок, а некоторые предоставляются за деньги и не допускают возможность редактирования ядра «движка».

До сих пор нет единой и чёткой классификации, существующих CMS, однако это не мешает выделить лидеров в данной области [12].

Важное замечание: бесплатность CMS отнюдь не означает низкое качество кода или наличие ограничений в использовании. Наоборот, чаще всего, отрытые и бесплатные CMS во многом выигрывают у своих коммерческих собратьев именно из-за своей общедоступности. Над этими системами трудятся не только коллективы профессиональных разработчиков, но и сотни и даже тысячи добровольцев, которые стремятся сделать CMS ещё лучше и безопасней.

Сложно сделать правильный выбор среди такого разнообразия, поэтому необходимо чётко осознавать целевое назначение будущего сайта. В таблице 1 представлено сравнение некоторых популярных CMS.

Таблица 1 – Обзор CMS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Общая информация** | [**AltoCMS**](http://www.cmsmagazine.ru/catalogue/altocms/) | [**Joomla!**](http://joomla.cmsmagazine.ru/) | [**MODX**](http://modx.cmsmagazine.ru/) | [**Magento**](http://www.cmsmagazine.ru/catalogue/magento/) | [**WordPress**](http://wordpress.cmsmagazine.ru/) |
| Лицензия | http://www.cmsmagazine.ru/img/s.gifБесплатная | http://www.cmsmagazine.ru/img/s.gifБесплатная | http://www.cmsmagazine.ru/img/s.gifБесплатная | http://www.cmsmagazine.ru/img/s.gifБесплатная | http://www.cmsmagazine.ru/img/s.gifБесплатная |
| Текущая версия | http://www.cmsmagazine.ru/img/s.gif1.0.10 | 3.6.2 | http://www.cmsmagazine.ru/img/s.gif1.0.12, Revolution 2.2.10-pl | http://www.cmsmagazine.ru/img/s.gif1.3.2.4 | http://www.cmsmagazine.ru/img/s.gif2.7.1 |
| Интерфейс администрирования | http://www.cmsmagazine.ru/img/s.gifWeb | http://www.cmsmagazine.ru/img/s.gifWeb | http://www.cmsmagazine.ru/img/s.gifWeb | http://www.cmsmagazine.ru/img/s.gifWeb | http://www.cmsmagazine.ru/img/s.gifWeb |
| База данных | MySQL, Postgres SQL | MySQL, MS SQL, Postgres SQL, InnerDB | http://www.cmsmagazine.ru/img/s.gifMySQL | http://www.cmsmagazine.ru/img/s.gifMySQL | http://www.cmsmagazine.ru/img/s.gifMySQL |
| Веб-сервер | http://www.cmsmagazine.ru/img/s.gif Apache | http://www.cmsmagazine.ru/img/s.gifApache, IIS, nginx | http://www.cmsmagazine.ru/img/s.gifApache, IIS | http://www.cmsmagazine.ru/img/s.gifApache | http://www.cmsmagazine.ru/img/s.gifApache |
| Языки программирования | http://www.cmsmagazine.ru/img/s.gifPHP | http://www.cmsmagazine.ru/img/s.gifPHP | http://www.cmsmagazine.ru/img/s.gifPHP | http://www.cmsmagazine.ru/img/s.gifPHP | http://www.cmsmagazine.ru/img/s.gifPHP |
| Фиксация действий пользователей | http://www.cmsmagazine.ru/img/s.gif- | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif- |
| Защита от автоматического заполнения форм (CAPTCHA) | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif- |
| Многоуровневое утверждение изменений | http://www.cmsmagazine.ru/img/s.gif- | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif- | http://www.cmsmagazine.ru/img/s.gif+ |
| Разграничение доступа | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ |
| Поддержка безопасного протокола при работе с системой (SSL) | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ |
| Руководство пользователя | http://www.cmsmagazine.ru/img/s.gif- | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ |
| Руководство разработчика | http://www.cmsmagazine.ru/img/s.gif- | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ |
| Дружественные URL (ЧПУ) | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ |
| Клонирование объектов системы (копирование материалов) | http://www.cmsmagazine.ru/img/s.gif- | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif- |
| Настройка стилей для визуального редактирования материалов | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif- |
| Восстановление удаленных объектов (Корзина) | http://www.cmsmagazine.ru/img/s.gif- | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif- | http://www.cmsmagazine.ru/img/s.gif- |
| Визуальный редактор (WYSIWYG-редактор) | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ |
| Публичный форум | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif- | http://www.cmsmagazine.ru/img/s.gif+ |
| Планировщик состояния материала | http://www.cmsmagazine.ru/img/s.gif- | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ | http://www.cmsmagazine.ru/img/s.gif+ |

1. Фреймворк для управления содержимым

Фреймворк для управления содержимым (англ. Content Management Framework, CMF) — это [каркас](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA) для проектирования систем управления контентом. На их основе создаются [системы управления содержимым](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D1%81%D0%BE%D0%B4%D0%B5%D1%80%D0%B6%D0%B8%D0%BC%D1%8B%D0%BC) (CMS), а также [веб-приложения](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5).

Если основная задача универсальных CMS — простота создания сайта без вмешательства программиста, то есть — конструктор сайтов, то CMF — это конструктор CMS (в том числе узкоспециализированных) для программиста [13].

Благодаря такому подходу сайт, созданный с помощью CMF, по сравнению с сайтом на базе CMS, может иметь более простую и безопасную в работе административную панель и быть менее требовательным к ресурсам системы.

При наличии грамотно спроектированной CMF и готовых модулей, программисту остаётся только сборка этих модулей в единое пространство.

Сложности могут возникнуть, если программист-пользователь фреймворка столкнётся с [программными ошибками](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B3) в его реализации.

Фреймворки веб-приложений обычно обеспечивают механизмы доступа к базе данных, [шаблонизации](https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D0%B8%D0%B7%D0%B0%D1%82%D0%BE%D1%80) и управление [сессиями](https://en.wikipedia.org/wiki/Session_(computer_science)) и т.д.

Большинство современных CMF являются реализацией архитектуры Model-View-Controller. Веб-фреймворк обеспечивает бесшовную интеграцию всех трёх слоёв MVC-архитектуры. Фреймворки скрывают от программиста детали, позволяя программисту сконцентрироваться на реализации [бизнес-логики](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%81-%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0)[[2]](https://ru.wikipedia.org/wiki/Content_Management_Framework#cite_note-TeamDevel-2).

Ряд [CMS](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D1%81%D0%BE%D0%B4%D0%B5%D1%80%D0%B6%D0%B8%D0%BC%D1%8B%D0%BC), предоставляющих [API](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) для расширения спектра своих функций, претендуют на звание CMF, хотя провести чёткую границу между CMS и CMF порой сложно. Основным отличием является то что на CMS можно построить сайт, не написав ни одной строчки кода.

Такая готовая к использованию система управления контентом, как [Drupal](https://ru.wikipedia.org/wiki/Drupal), одновременно считается и каркасом для построения таких систем[[6]](https://ru.wikipedia.org/wiki/Content_Management_Framework#cite_note-6), что определяется как возможностью расширения спектра предоставляемых функций за счёт пользовательских модулей, так и богатством механизмов и абстракций для управления контентом, предоставляемым этой системой.

Примеры CMF приведены в таблице 2.

Таблица 2 – Примеры CMF

|  |  |  |
| --- | --- | --- |
| **Название** | **Платформа** | **Поддерживаемые СУБД** |
| CakePHP | PHP | PostgreSQL, MySQL, SQLite, MS SQL, Oracle |
| Catalyst | Perl | PostgreSQL, MySQL, SQLite, MS SQL, Oracle |
| ZendFramework | PHP | PostgreSQL,MariaDB, MySQL, SQLite, MS SQL, Oracle |
| Ruby on Rails | Ruby | MySQL, PostgreSQL, SQLite, Firebird, Oracle, SQL Server, DB2 |
| Django | Python | PostgreSQL,MariaDB, MySQL, SQLite, |

Нередко, CMF-системы обладающие достаточным набором функций для разработки полноценных веб-приложений и легких в управлении сайтов различных типов, называют CMF/CMS системами, поскольку они объединяют в себе возможности обеих видов систем [11]. Примеры CMF/CMS систем представлены в таблице 3:

Таблица 3 – Примеры CMS/CMF

|  |  |  |
| --- | --- | --- |
| **Название** | **Платформа** | **Поддерживаемые СУБД** |
| MODx | PHP | MySQL |
| Drupal | PHP | MySQL, PostgreSQL |
| eZ publish | PHP | MySQL, PostgreSQL |
| TYPO3 | PHP | MySQL, PostgreSQL |

В CMF/CMS системах можно выделить несколько характерных особенностей:

1. Детерминированная внутренняя архитектура. В CMF/CMS системах внутренняя архитектура имеет развитые механизмы абстракции, не зависящие от CMS-образующих модулей. Это значит, что сопровождать проект, выполненный на основе CMF/CMS гораздо проще проекта, сделанного на «чистой» CMS.
2. Многофункциональность. Веб-сайты и веб-приложения, выполненные на основе CMF/CMS обладают высокой степенью индивидуализации: каждый проект может быть адаптирован применительно к конкретной ситуации. Многофункциональность CMF/CMS систем позволяет создавать на их основе любые интернет-проекты, от небольших сайтов-визиток до разветвленных порталов или интернет-магазинов.
3. Расширяемость и совместимость. Существующий набор функций CMF/CMS систем может быть расширен за счет интеграции дополнительных модулей и программных кодов. Большинство CMF/CMS поддерживают работу с различными СУБД (MySQL, Oracle, PostgreSQL и др.), выполняют трансляцию данных в любой требуемый формат (XHTML, JSON-структуры, PDF, XLS, RTF и т.д.).
4. Шаблонизация. В CMF/CMS поддерживается шаблонизация — удобное создание, а также интеграция шаблонов представления (дизайна) без необходимости затрагивать программное ядро или содержимое разрабатываемого сайта.
5. Удобство использования. CMF/CMS могут представлять собой как системы с произвольным, достаточно расширяемым и настраиваемым набором функциональных возможностей, так и дистрибутивы, удобные для использования непрофессиональными программистами [14].

Владельцы веб-сайтов на CMF/CMS получают систему настройки web-интерфейса, визуальный редактор для наполнения контентом, систему хранения и получения информации, систему упорядочивания информации, систему управления пользователями — т. е. все возможности, характерные для CMS.

В ходе проведенной работы была создана классификация средств разработки сайтов по критерию причастности программиста к созданию сайта, которая представлена на рис. 1.

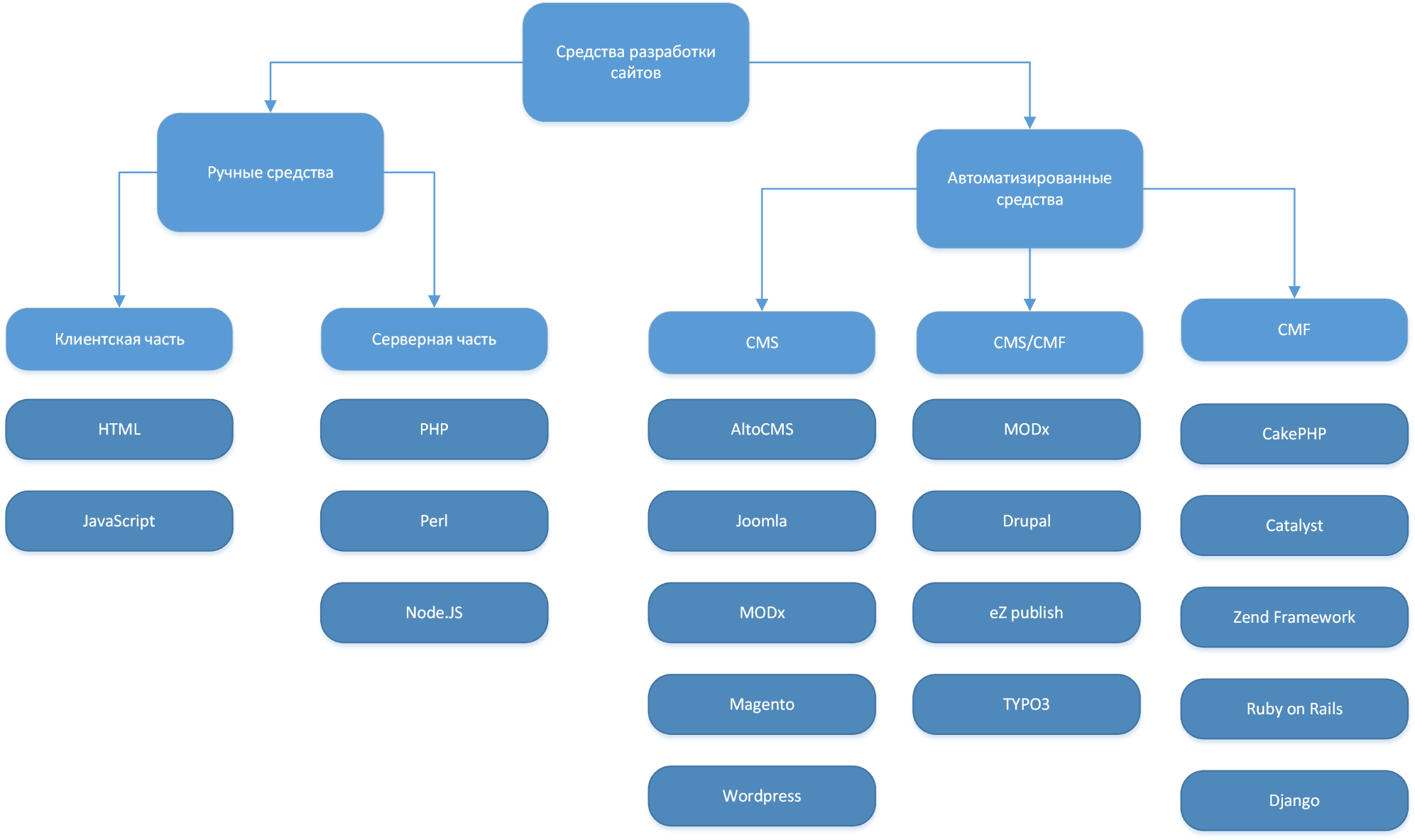


Рисунок 1 – Классификация средств разработки сайтов по степени причастности разработчика

Заключение

В сегодняшних условиях у разработчика есть множество способов создания сайтов. Выбор тех или иных средств определяется множеством факторов, таких как порог вхождения разработчика, для каких нужд создается сайт, техническое задание на выполняемый проект и т.д.

В ходе работы были рассмотрены понятия сайта, web-страницы, были рассмотрены типы web-страниц, была разработана классификация средств разработки сайтов по критерию степени причастности разработчика.

В ходе работы были рассмотрены наиболее используемые языки программирования для написания сайтов, которые были объединены в понятие «ручные средства разработки сайтов». Также были рассмотрены автоматизированные средства, такие как системы управления контентом и фреймворк для управления содержимым, а также их комбинации с кратким анализом этих средств.

В дальнейшем планируется более подробно рассмотреть понятие «целевое назначение сайта», рассмотреть языки программирования, которые используются только для web-программирования и языки, которые могут, кроме web, использоваться для других задач, а также создать классификацию средств разработки сайтов по их целевому назначению.

Список использованных источников

1. Ford S., Wells D., Wells N. Web Programming Languages (1 September 1997) [электронный ресурс], URL - http://www.objs.com/survey/lang.htm (дата обращения 10.10.2017)
2. Эд Титтел, Джефф Ноубл. HTML, XHTML и CSS для чайников, 7-е издание = HTML, XHTML & CSS For Dummies, 7th Edition. — М.: «Диалектика», 2011. — 400 с.
3. Стивен Шафер. HTML, XHTML и CSS. Библия пользователя, 5-е издание = HTML, XHTML, and CSS Bible, 5th Edition. — М.: «Диалектика», 2010. — 656 с.
4. TC39. 4. Overview // ECMAScript Language Specification. — 5th ed. — 2009. — P. 1, 2. http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf
5. Флэнаган Д. JavaScript. Карманный справочник. Сделайте веб-страницы интерактивными! / Перевод А.Г. Сысонюк. — Москва.: Издательский дом "Вильямс", 2015. — С. 320
6. Джейсон Ленгсторф. PHP и jQuery для профессионалов = Pro PHP and jQuery. — М.: «Вильямс», 2010. — С. 352
7. Кузнецов Максим, Симдянов Игорь. PHP на примерах. — 2-е изд. перераб. и доп. — СПб.: «БХВ-Петербург», 2011. — С. 400.
8. Рэндал Л. Шварц, Том Феникс, Брайан Д. Фой. Изучаем Perl. — М.: O’Reilly, «Символ», 2009. — С. 377
9. Кристиансен, Торкингтон. Perl. Сборник рецептов. Для профессионалов = Perl Cookbook. — М.: O’Reilly, «Питер», 2004. — 928 с
10. Итан Браун. Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript = Web Development with Node and Express / Итан Браун. — Санкт-Петербург: Питер, 2017. — 336 с.
11. Martin Brampton. PHP5 CMS Framework Development. — PACKT publishing, June 2008. — 328 p.
12. Савельева Н. Системы управления контентом // Открытые системы. — 2004. — № 4.
13. Сычев И.А. Создание сайтов на основе систем управления контентом: электрон. учеб.-метод. пособие / И.А. Сычев, В.Н. Половников. Бийск: АГАО, 2012
14. Декстер М. Joomla!: программирование / М. Декстер, Л. Лэндри; пер. с англ. и ред. И.В. Берштейна. М.: Вильямс, 2013.