|  |  |
| --- | --- |
| *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»)** |
|  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Факультет |  | И |  | «Информационные и управляющие системы» |
|  |  | шифр |  | Наименование |
| Кафедра |  | И5 |  | «Информационные системы и программная инженерия» |
|  |  | шифр |  | наименование |
| Дисциплина |  | Научно-исследовательская работа в семестре | | |

ОТЧЕТ

на тему

|  |
| --- |
| Семантическое ядро сайта. Классификация систем |
| управления содержимым по способу создания |
| страницы |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил магистрант группы | | | |  | | И9М33 |
| Гаврютина А.А. | | | | | | |
| Фамилия И.О. | | | | | | |
| **ПРЕПОДАВАТЕЛЬ** | | | | | | |
| Каминский В.Н. | |  |  | | | |
| Фамилия И.О. Подпись | | | | | | |
| Оценка |  | | | |  | |
| «\_\_\_\_\_» |  | | | | 2018 г. | |

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc532159365)

[1 СЕМАНТИКА ВЕБ-САЙТА. СЕМАНТИЧЕСКОЕ ЯДРО 5](#_Toc532159366)

[2 КЛАССИФИКАЦИЯ СИСТЕМ УПРАВЛЕНИЯ СОДЕРЖИМЫМ ПО СПОСОБУ СОЗДАНИЯ СТРАНИЦЫ 8](#_Toc532159367)

[3 ФОРМИРОВАНИЕ ОТОБРАЖЕНИЯ ВЕБ-СТРАНИЦЫ 13](#_Toc532159368)

[ЗАКЛЮЧЕНИЕ 18](#_Toc532159369)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 19](#_Toc532159370)

ВВЕДЕНИЕ

Для выполнения работы были поставлены следующие задачи:

* Ввести понятие «семантическое ядро сайта».
* Рассмотреть классификации средств управления содержимым по способу формирования страницы.
* Рассмотреть способ формирования веб-браузером отображения веб-страницы.

1 СЕМАНТИКА ВЕБ-САЙТА. СЕМАНТИЧЕСКОЕ ЯДРО

Основной целью посещения сайтов в Интернете является поиск информации. Существует два способа получения необходимой информации: вы можете ввести URL-адрес в браузере или ввести запрос в систему поиска информации, а затем выбрать нужный сайт из предложенного. Второй путь становится все более популярным.

Как сделать сайт включенным в список предлагаемых сайтов? Более того, желательно, чтобы сайт был одним из первых в этом списке.

Прежде всего, нужно сформировать семантическое ядро ​​сайта.

Семантическое ядро ​​сайта – это список так называемых ключевых слов, которые наиболее точно определяют тему сайта и которые в основном используются для поиска товаров или услуг. Формирование этого списка имеет большое значение для дальнейшего продвижения сайта.

Алгоритм формирования семантического ядра сайта можно описать следующим образом:

1. Учитывая целевое назначение сайта, необходимо составить список ключевых слов. Количество этих слов не ограничено. Базовый список необходимо расширить теми запросами, которые вводили пользователи в строке поиска наряду со словами из базового списка. Найти такие слова и фразы можно с помощью сервиса статистики «Яндекс.Вордстат». По каждому отдельному «ключевику» сервис покажет варианты запросов, связанные с ним. Данная информация доступна в левой колонке Wordstat. В правой колонке Wordstat будут показаны запросы, похожие на исходное ключевое слово. Многие из них также можно включить в семантическое ядро, если они имеют отношение к тематике сайта.

Обязательно стоит учитывать частотность запросов «Вордстат», применяя специальные поисковые операторы для уточнения статистики. Среди собранных слов и фраз могут быть «нулевые» — с очень низкой частотой запросов в месяц (например, 5 и меньше), которые не способны приводить трафик на сайт [1].

2. Когда список ключевых слов составлен, нужно проверить статистику запросов ключевых слов.  Для этого существует множество инструментов, к примеру, Тенденции Google.

3. Следующим шагом будет написать тексты для сайта, используя выбранные слова: с помощью метатегов, описание с использованием метатегов в заголовках.

4. В процессе сопровождения, требуется отслеживать статистику посещаемости сайта, оптимизировать тексты и т.д.

При правильном формировании семантического ядра ​​ сайта, возможно будет определить размер целевой аудитории и, следовательно, потенциальную посещаемость сайта, которая позже предоставит дальнейшее успешное продвижение сайта в информационно-поисковых системах [2].

При составлении семантического ядра сайта требуется учитывать следующие факторы:

1. Количество показов или частотность. Запросы делятся на высокочастотные (ВЧ), среднечастотные (СЧ) и низкочастотные (НЧ).
2. Количество показов запроса без словосочетаний, в которые он входит.
3. Количество показов без использования морфологических форм (морфологическая форма — это любое морфологическое изменение — склонение, спряжение — запроса).
4. Релевантные поисковым запросам страницы.
5. Оценка стоимости продвижения по конкретному запросу.
6. Конкурентность запроса. Определяется по наличию в ТОПе сайтов, продвигаемых по этому запросу. Различают высококонкурентные (ВК), среднеконкурентные (СК) и низкоконкурентные (НК) запросы.
7. Прогнозируемое и реальное количество переходов, составляемое на основе анализа ТОПа и количества показов запроса без использования морфологических форм.
8. Показатель отказов (количество посетителей, просмотревших не более одной страницы). Определяется для каждого потенциального запроса отдельно, на основе статистических данных сайта.
9. Сезонность и геозависимость запроса. Различные товары и услуги, ради которых создан ресурс, могут обладать разным спросом в зависимости от сезона и географического положения потенциальных потребителей.
10. Иные факторы, относящиеся к ключевым словам.

Сформированное семантическое ядро позволяет точно определить тематику сайта, а, следовательно, и его целевое назначение. Под целевым назначением будем понимать то содержимое, которое мы хотим дать пользователю для удовлетворения его потребностей. После определения целевого назначения сайта можно приступать к его разработке. Процесс разработки включает в себя проектирование средств человеко-машинного взаимодействия для клиентской части, выбор средств разработки серверной и клиентской частей веб-сайта, и непосредственно написание исходных текстов программ.

Но хочется отметить, что напрямую семантику сайта и выбор средств разработки веб-сайта нельзя, т.к. семантическое ядро представляет собой набор слов, описывающих некую тематику, но оно не дает полного представления о целевом назначении сайта.

2 КЛАССИФИКАЦИЯ СИСТЕМ УПРАВЛЕНИЯ СОДЕРЖИМЫМ ПО СПОСОБУ СОЗДАНИЯ СТРАНИЦЫ

Под автоматизированным способом разработки подразумевается 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).

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 [3].

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 [4].

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.

Существует множество готовых систем управления содержимым сайта, в том числе и свободно распространяемых.

Их можно разделить на три типа по способу создания страницы:

1. Генерация страниц по запросу.

Системы такого типа работают на основе связки «Модуль редактирования → База данных → Модуль представления». Модуль представления генерирует страницу с содержанием при запросе на него, на основе информации из базы данных. Информация в базе данных изменяется с помощью модуля редактирования. Страницы заново создаются сервером при каждом запросе, что в свою очередь создаёт дополнительную нагрузку на системные ресурсы. Нагрузка может быть многократно снижена при использовании средств кэширования, которые имеются в современных веб-серверах [5].

1. Генерация страниц при редактировании.

Системы этого типа служат для редактирования страниц, которые при внесении изменений в содержание сайта создают набор статических страниц. При таком способе в жертву приносится интерактивность между посетителем и содержимым сайта.

1. Смешанный тип.

Данный тип сочетает в себе преимущества первых двух. Может быть реализован путём кэширования — модуль представления генерирует страницу один раз, в дальнейшем она в несколько раз быстрее подгружается из кэша. Кэш может обновляться как автоматически, по истечении некоторого срока времени или при внесении изменений в определённые разделы сайта, так и вручную по команде администратора. Другой подход — сохранение определённых информационных блоков на этапе редактирования сайта и сборка страницы из этих блоков при запросе соответствующей страницы пользователем.

Также можно предложить следующую классификацию CMS по принципу создания веб-страниц:

1. Полностью динамические системы

Полностью динамические системы создают запрошенные документы динамически каждый раз, когда они вызываются, то есть шаблоны и содержимое интерпретируются или объединяются только при извлечении и выводе. Преимущество этих систем в том, что страница всегда «актуальна». К недостаткам можно отнести то, что пересоздание каждой страницы может привести к задержке доставки страниц или недостаточному оборудованию / конфигурации вычислительной мощности по отношению к количеству одновременных пользователей на серверную перегрузку и в крайних случаях к простою системы при большой нагрузке (например, большое количество посетителей). Одной из мер по снижению нагрузки является [кэширование](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com&sl=auto&sp=nmt4&tl=ru&u=https://de.wikipedia.org/wiki/Cache&xid=17259,15700021,15700124,15700149,15700186,15700191,15700201,15700237,15700242&usg=ALkJrhhETA9IF0n9ow0UoBUqKbkJNqhjnQ) [4].

1. Статические системы

Статические системы генерируют отдельные веб-страницы из шаблонов и содержимого в виде статически сохраненного файла в [файловой системе](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com&sl=auto&sp=nmt4&tl=ru&u=https://de.wikipedia.org/wiki/Dateisystem&xid=17259,15700021,15700124,15700149,15700186,15700191,15700201,15700237,15700242&usg=ALkJrhga0ELEs_FJ5ElvkPP1NJk-xeYhNg) или, возможно, в базе данных. Таким образом, конечный продукт получил документы, которые не интерпретируются серверной технологией как ASP, JSP или PHP и, следовательно, могут быть выведены непосредственно через веб-сервер, что отражается на скорости вывода. Это имеет то преимущество, что в качестве основы может быть достаточно даже более простых продуктов для веб-хостинга. Чисто статические системы формируют историческое происхождение CMS, но - в своем первоначальном виде - используются крайне редко.

1. Гибридные системы

Гибридные системы сочетают в себе преимущества статической и полностью динамической генерации страниц. Только содержимое, которое должно динамически генерироваться из базы данных (например, новости, поисковые запросы, персонализированный контент или данные магазина), считываются из базы данных во время выполнения. Все остальное содержимое, которое не подвержено постоянным изменениям (например, структура страницы, навигация, а также определенные тексты и изображения), является статическим [3].

1. Полустатические системы

Полустатические системы генерируют контент таким образом, чтобы он был статическим, но в то же время динамичным, т.е., все данные хранятся непосредственно в статически сформированных файлах, которые затем выводятся немедленно по требованию. Динамическое содержимое генерируется, когда код на языке программы интегрирован в файл или отдельные [записи данных](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com&sl=auto&sp=nmt4&tl=ru&u=https://de.wikipedia.org/wiki/Datensatz&xid=17259,15700021,15700124,15700149,15700186,15700191,15700201,15700237,15700242&usg=ALkJrhisM55Ret4bM7G74TREmdAAvGVFGQ) изменены или созданы заново.

3 ФОРМИРОВАНИЕ ОТОБРАЖЕНИЯ ВЕБ-СТРАНИЦЫ

Основное предназначение браузера – отображать веб-ресурсы. Для этого на сервер отправляется запрос, а результат выводится в окне браузера. Под ресурсами в основном подразумеваются HTML-документы, однако это также может быть PDF-файл, картинка или иное содержание. Расположение ресурса определяется с помощью URI (унифицированного идентификатора ресурсов).

То, каким образом браузер обрабатывает и отображает HTML-файлы, определено спецификациями HTML и CSS. Они разрабатываются Консорциумом W3C, который внедряет стандарты для Интернета.   
Многие годы браузеры отвечали лишь части спецификаций, и для них создавались отдельные расширения. Для веб-разработчиков это означало серьезные проблемы с совместимостью. Сегодня большинство браузеров в большей или меньшей степени отвечает всем спецификациям.

Пользовательские интерфейсы разных браузеров имеют много общего. Основные элементы интерфейса браузера перечислены ниже.

1. Адресная строка для ввода URI.
2. Кнопки навигации «Назад» и «Вперед».
3. Закладки.
4. Кнопки обновления и остановки загрузки страницы.
5. Кнопка «Домой» для перехода на главную страницу.

Как ни странно, спецификации, которая бы определяла стандарты пользовательского интерфейса браузера, не существует. Современные интерфейсы являются результатом многолетней эволюции, а также того, что разработчики частично копируют друг друга. В спецификации HTML5 не указано, что именно должен содержать интерфейс браузера, однако перечислены некоторые основные элементы. К ним относится адресная строка, строка состояния и панель инструментов. Разумеется, существуют и специфические функции, такие как менеджер загрузок в Firefox.

Ниже перечислены основные компоненты браузера (см. рисунок 1).

**Пользовательский интерфейс** – включает адресную строку, кнопки «Назад» и «Вперед», меню закладок и т. д. К нему относятся все элементы, кроме окна, в котором отображается запрашиваемая страница.

**Механизм браузера** – управляет взаимодействием интерфейса и модуля отображения.

**Модуль отображения** – отвечает за вывод запрошенного содержания на экран. Например, если запрашивается HTML-документ, модуль отображения выполняет синтаксический анализ кода HTML и CSS и выводит результат на экран.

**Сетевые компоненты** – предназначены для выполнения сетевых вызовов, таких как HTTP-запросы. Их интерфейс не зависит от типа платформы, для каждого из которых есть собственные реализации.

**Исполнительная часть пользовательского интерфейса** – используется для отрисовки основных виджетов, таких как окна и поля со списками. Ее универсальный интерфейс также не зависит от типа платформы. Исполнительная часть всегда применяет методы пользовательского интерфейса конкретной операционной системы.

**Интерпретатор JavaScript** – используется для синтаксического анализа и выполнения кода JavaScript.

**Хранилище данных** – необходимо для сохраняемости процессов. Браузер сохраняет на жесткий диск данные различных типов, например, файлы cookie. В новой спецификации HTML (HTML5) имеется определение термина "веб-база данных": это полноценная (хотя и облегченная) браузерная база данных.

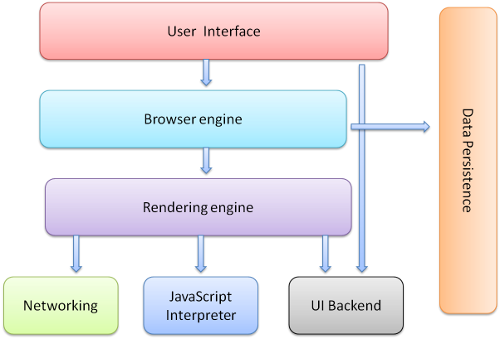


Рисунок 1 – Основные компоненты браузера [6]

Следует отметить, что Chrome, в отличие от большинства браузеров, использует несколько экземпляров модуля отображения, по одному в каждой вкладке, которые представляют собой отдельные процессы.

**Модуль отображения**

Как можно догадаться по названию, модуль отображения отвечает за вывод запрошенного содержания на экране браузера.

По умолчанию он способен отображать HTML- и XML-документы, а также картинки. Специальные подключаемые модули (расширения для браузеров) делают возможным отображение другого содержания, например, PDF-файлов.

**Модули отображения**

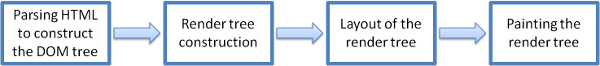
В браузерах Firefox, Chrome и Safari используются два модуля отображения. В Firefox применяется Gecko – собственная разработка Mozilla, а в Safari и Chrome используется WebKit.

WebKit представляет собой модуль отображения с открытым исходным кодом, который был изначально разработан для платформы Linux и адаптирован компанией Apple для Mac OS и Windows. Подробные сведения можно найти на сайте webkit.org.

**Основная схема работы**

Модуль отображения получает содержание запрошенного документа по протоколу сетевого уровня, обычно фрагментами по 8 КБ.

Схема дальнейшей работы модуля отображения выглядит приведенным ниже образом. (см. рисунок 2).

Рисунок 2 – Схема работы модуля отображения [6]

Модуль отображения выполняет синтаксический анализ HTML-документа и переводит теги в узлы DOM в дереве содержания. Информация о стилях извлекается как из внешних CSS-файлов, так и из элементов style. Эта информация и инструкции по отображению в HTML-файле используются для создания еще одного дерева – дерева отображения.

Оно содержит прямоугольники с визуальными атрибутами, такими как цвет и размер. Прямоугольники располагаются в том порядке, в каком они должны быть выведены на экран.

После создания дерева отображения начинается компоновка элементов, в ходе которой каждому узлу присваиваются координаты точки на экране, где он должен появиться. Затем выполняется отрисовка, при которой узлы дерева отображения последовательно отрисовываются с помощью исполнительной части пользовательского интерфейса.

Важно понимать, что это последовательный процесс. Для удобства пользователя модуль отображения старается вывести содержание на экран как можно скорее, поэтому создание дерева отображения и компоновка могут начаться еще до завершения синтаксического анализа кода HTML. Одни части документа анализируются и выводятся на экран, в то время как другие только передаются по сети [7].

Как видно из рисунков 3 и 4, в WebKit и Gecko используется разная терминология, однако схемы их работы практически идентичны.

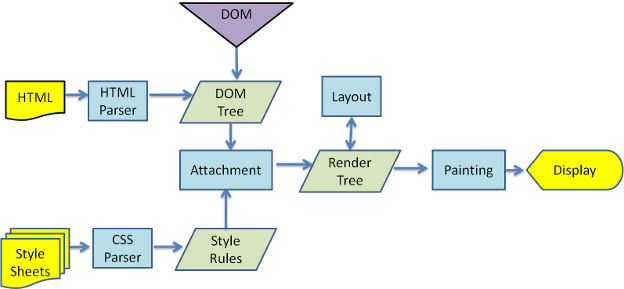


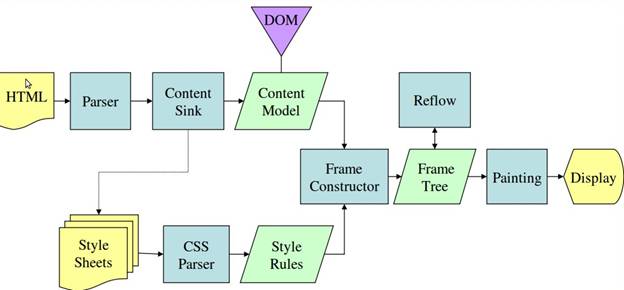
Рисунок 3 – Схема работы модуля отображения WebKit

Рисунок 4 – Схема работы модуля отображения Mozilla Gecko

В Gecko дерево визуально отформатированных элементов называется деревом фреймов (frame tree), в котором каждый элемент является фреймом. В WebKit используется дерево отображения (render tree), состоящие из объектов отображения (render objects).

Размещение элементов в WebKit называется компоновкой, или версткой (layout), а в Gecko – обтеканием (reflow). Объединение узлов DOM и визуальных атрибутов для создания дерева отображения называется в WebKit совмещением (attachment). Небольшое отличие Gecko, не имеющее отношения к семантике, состоит в том, что между HTML-файлом и деревом DOM находится еще один уровень.

Он называется буфером содержания (content sink) и служит для формирования элементов DOM.

ЗАКЛЮЧЕНИЕ

В ходе работы были рассмотрены понятия сематического ядра сайта, были рассмотрены классификации средств управления содержимым, а также способы отображения веб-страницы веб-браузером.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Дари Кристиан, Сирович Джейми. Поисковая оптимизация сайта (SEO) на ASP.NET для профессионалов. Руководство разработчика по SEO = Professional Search Engine Optimization with ASP.NET: A Developer's Guide to SEO. — М.: «Диалектика», 2008. — 400 с.
2. Кент Питер. Поисковая оптимизация для чайников, 4-е издание = Search Engine Optimization For Dummies, 4th Edition. — М.: «Вильямс», 2011. — 432 с.
3. Mauthe Andreas, Thomas Peter. Professional Content Management Systems: Handling Digital Media Assets, John Wiley & Sons, 2008. ­­­– 330 p.
4. Krüger Jörg Dennis, Kopp Matthias. Web Content managen. Professioneller Einsatz von Content-Management-System. Markt & Technik, 2002. – 336 p.
5. Martin Brampton. PHP5 CMS Framework Development. — PACKT publishing, June 2008. — 328 p.
6. Grosskurth, Alan. A Reference Architecture for Web Browsers // ICSM '05 Proceedings of the 21st IEEE International Conference on Software Maintenance, pp. 661-664
7. Стивен Шафер. HTML, XHTML и CSS. Библия пользователя, 5-е издание = HTML, XHTML, and CSS Bible, 5th Edition. — М.: «Диалектика», 2010. — 656 с.