Содержание:

ВВЕДЕНИЕ

Гипертекст является своеобразным элементом программирования, необходимым для соединения разных массивов данных, в чем и заключается его значение.

В то же время статический гипертекст помогает пользователям Сети ориентироваться на определенных сайтах, выступает в роли навигатора, помогает облегчить диалог с информационными ресурсами.

При использовании базы данных гипертекст также становится динамичным и помогает связать сведения из других Интернет – ресурсов.

С целью удобного преподнесения гипертекста в сети необходимо именно язык HTML, особенности применения которого будут рассмотрены в курсовой работе.

Целью написания курсовой работы является изучение специфики использования языков гипертекстовой разметки и, в частности, языка HTML.

Для реализации цели автором поставлены следующие задачи:

- рассмотреть сущность и развитие языка HTML;

- предоставить сравнительную характеристику языков HTML и XHTML;

- изучить методику работы с языком HTML;

- ознакомиться с приемами обработки тэгов при помощи языка HTML.

Предметом написания курсовой работы являются языки гипертекстовой разметки, а объектом исследования выбраны специфические особенности языка HTML.

Структура курсовой работы представлена введением, основной частью, состоящей из двух разделов, а также заключения.

Завершает работу список используемой литературы.

1 ОСОБЕННОСТИ ЯЗЫКОВ ГИПЕРТЕКСТОВОЙ РАЗМЕТКИ И ИХ СТРУКТУРА

1.1 История создания и развития HTML

За появление HTML современный мир должен благодарить одного ученого Европейского совета по ядерным исследованиям (Conseil Européen pour la Recherche Nucléaire, CERN). Зовут этого ученого Тимоти Джон Бернерс-Ли. Первая версия HTML создавалась для целей форматирования научных документов. Именно структурного форматирования без элементов описания цветовых схем, параметров шрифта и т.п. Таким образом, изначально HTML позволял выделять в тексте заголовки, абзацы, списки и им подобные структурные элементы. Результат обработки или “воспроизведения” HTML не должен был зависеть от технических особенностей аппаратных средств его визуализации, поскольку не содержал в себе параметров этой визуализации. Со временем такая особенность языка разметки гипертекста была частично утрачена.

Итак, появление первых версий HTML относят к 1986 году, а в 1991 году HTML был существенно доработан и стал использоваться именно для передачи гипертекста по просторам всемирной паутины. Говорят, что всемирно известная аббревиатура HTML, расшифровывающаяся как Hyper Text Markup Language (язык разметки гипертекста) появилась как раз в начале 90-х годов прошлого века. А теперь небольшой экскурс в родословную языков разметки. Первая версия языка разметки гипертекста HTML была создана на основе стандарта обобщенного языка разметки SGML (Standard Generalized Markup Language), который в некотором роде можно считать прообразом расширяемого языка разметки данныхXML (eXtensible Markup Language). Стандарт XML в наше время приобрел огромную популярность благодаря большому количеству своих расширений, используемых в компьютерных технологиях. Чтобы совсем запутать читателя сразу добавлю, что впоследствии на основе XML был разработан язык разметки гипертекста XHTML (Extensible Hypertext Markup Language), повторяющий по своей сути HTML. В результате мы имеем аббревиатуры SGML, HTML, XML и XHTML, и необходимо понять, кто из них кто. На самом деле все просто: SGML это не что иное, как набор правил, на основе которых можно строить любые языки разметки. HTML и есть один из этих языков - приложение SGML. Другими словами, SGML определяет то, как должны выглядеть элементы разметки, а HTML - какие именно должны быть элементы и как они

должны интерпретироваться браузерами. XHTML, в свою очередь, является приложением XML, а сам XML ни что иное, как упрощенный вариант SGML. Языки HTML и XHTML, не смотря на то, что очень внешне похожи, имеют существенные скрытые отличия, которые, по большей части, заключаются в принципе их обработки.

Теперь вернемся к истории развития HTML. Итак, до 1994 года HTML по-прежнему использовался только для структурной разметки данных, хотя в его составе уже появились теги для выделения текста жирным или курсивом. В том же 1994 году создается организация W3C (World Wide Web Consortium) – Консорциум всемирной паутины, которую возглавляет, что вполне логично, тот самый Тим Бернерс-Ли, и в 1995 году в свет выходит рекомендация HTML 2.0. Создатели HTML уже тогда понимали, что со временем их детище из языка статичной разметки текста эволюционирует в основной инструмент создания динамических интернет ресурсов. Главным дополнением HTML 2.0 стало появление в составе языка форм с наборами пользовательских элементов управления, которые должны были использоваться для ввода пользователем параметров HTTP запросов.

После выхода второй версии сразу же началась работа над следующим поколением HTML. В 1997 году выходит рекомендация HTML 3.2, которая дополнила язык разметки таблицами, фреймами, изображениями и некоторыми другими важными тегами. Но самым главным достижением 3-й версии является то, что ее авторы вновь вернулись к проблеме визуализации разметки в браузере, вспомнили про то, что HTML должен размечать лишь структуру документа и не должен содержать непосредственно в себе параметры графических стилей отображения элементов в браузере. Результатом их работ над HTML 3.2 стало появление самостоятельного языка CSS (Cascading Style Sheets) – каскадных таблиц стилей, код которого можно теперь подключать к коду разметки HTML и тем самым настраивать внешний вид страницы.

К выходу 4-й версии HTML в 1997 году сотрудники W3C избавили свое детище от тех ненужных элементов, которые с появлением CSS стали устаревшими и компрометировали идею отделения разметки структуры от параметризации представления. Но из-за таких мелочей никто не стал бы городить новую версию. Основное достижение рекомендаций HTML 4.0 – появление объектной модели страницы (Document Object Model, DOM), элементами которой теперь можно было манипулировать посредствам скриптовых языков программирования, исполняемых браузерами. Самым популярным таким языком программирования является JavaScript. HTML плюс DOM плюс JavaScript равно Dynamic HTML или просто DHTML,

который ознаменовал прорыв в веб-дизайне. Теперь элементы загруженной интернет страницы могли изменять свой внешний вид в ответ на действия пользователя, а также добавлять новые и удалять имеющиеся элементы. В 24.12.1999 году выходит последняя редакция 4-й версии языка разметки гипертекста – HTML 4.01.

Версия HTML5 до сих пор еще не получила статус официальных рекомендаций W3C, но уже сейчас понятно, что авторы HTML продолжают работать в направлении разработки требований к поддержке объектной модели документа и интерпретации JavaScript. Хотя HTML5 и получит ряд новых тегов, но большая часть рекомендаций все же касается поведения браузера в контексте работы DHTML: появится встроенная поддержка функций перетаскивания элементов (drag-and-drop), возможность рисовать на виртуальном полотне (canvas), управлять просмотром истории, обмениваться сообщениями между страницами, сохранять контекст исполнения и многое другое. Есть надежда, что с выходом новых рекомендаций HTML проблемы отсутствия кроссбраузерности, когда один и тот же JavaScript код исполняется под управлением разных браузеров по-разному, будут постепенно исчезать. Ведь тенденция определять требования к работе с объектной моделью и JavaScript будет сохраняться, а разработчики браузеров будут обязаны (если хотят, чтобы их программными продуктами пользовались) следовать этим требованиям.

Выход HTML5 запланирован на 2014 году. Возможно, к тому времени W3C разработает отдельные рекомендации, касающиеся только программирования на JavaScript, а HTML со временем снова станет исключительно языком разметки структуры документа. Не смотря на то, что сегодня еще только 2012 год, многиеновые возможности HTML5 уже сейчас поддерживаются наиболее популярными браузерами. Многое, что приходилось веб-дизайнерам раньше делать самостоятельно (тот же drag-and-drop), с выходом HTML5 будет поддерживаться на уровне браузера, и такой ход развития событий не может не радовать. Остается надеяться, что тенденция сохранится.

1.2 Сравнение стандартов HTML и XHTML

Итак, язык XHTML повторяет и дополняет функциональность HTML, а зачем он это делает, я как раз и попытаюсь изложить в этом разделе. Поскольку XHTML является расширением XML, то все требования к правильно сформированному (well-

formed) XML документу сохраняются. Вот те самые дополнительные требования к разметке документа, если он должен соответствовать стандарту XHTML:

- каждый тег XHTML должен быть закрыт. Если HTML позволял конструкции типа
 или <hr>, то в XHTML они должны выглядеть только так:
<hr/>. Менее тривиальным является следующий вариант разметки, который устраивает HTML, но не является корректным с точки зрения XML:

boldbold_and_italicitalic

Правильным XHTML аналогом будет являться следующая разметка

boldbold_and_italicitalic

На основе приведенного примера, первое ограничение я бы дополнил формулировкой: XHTML не допускает частичного пересечения области действий тегов разметки. Если это обстоятельство и создает какие-то дополнительные сложности верстальщикам, то эти сложности с лихвой компенсируются контролем над ошибками со стороны сервисов XML. Причины всех этих ограничений проявятся дальше.

HTML не поддерживает сокращенной формы записи атрибутов. Это означает, что в XHTML нет сокращенной формы записи *булевых атрибутов*, а само значение атрибутов всегда должно быть в кавычках. Если в HTML следующий код считался корректным

<input type=textbox readonly value='anytext'/>

, то в XHTML приведенная конструкция должна выглядеть следующим образом:

<input type='textbox' readonly='readonly' value='anytext'/>

Специальные символы в XHTML должны быть представлены в виде кодов. Это означает, например, что символы < и >, если они не являются частью разметки, должны в тексте обозначаться, как < и > соответственно. Если такой вариант не устраивает, например, если требуется в разметку добавить программный код (Java-script, VBScript), то для этих целей следует использовать раздел CDATA, содержимым которого может быть любая символьная информация, в том числе специальные символы разметки. Вот пример:

<script type="text/javascript">

<![CDATA[

Код скрипта

]]>

</script>

Все символы, используемые в именах тегов и атрибутов должны быть строчными. Вот это ограничение уже не является наследием XML, поскольку XML настаивает только на том, чтобы и открывающий и закрывающий теги были записаны одинаковым набором символов, в который могут входить как строчные, так и заглавные символы. Это ограничение, скорее, результат стремления избежать путаницы и оптимизировать скорость обработки документа. Кодировка символов в XHTML, как и в XML по умолчанию UTF-8.

Корневой элемент в XHTML должен быть один. Другими словами это означает, что тег HTML должен присутствовать в XHTML всегда! Стандарт HTML не настаивал на присутствии тегов <HTML> и <BODY> – разметку можно было начинать с любого тега и корневого элемента могло не быть вообще.

Теперь пару слов о менее очевидных отличиях между двумя этими стандартами. Повторюсь, что все “разногласия” между HTML и XHTML являются следствием того, что XHTML является расширением XML, а HTML нет.

Обработчик HTML является, по сути, интерпретатором. Он обрабатывает документ последовательно, и именно это обстоятельство позволяет ему исправлять ошибки разметки. Многим известно, что документ HTML в памяти браузера представлен в виде объектной модели DOM. Именно DOM является основой DHTML (Dynamic HTML) – симбиоза HTML и Java Script, который способен “оживлять” статичную разметку, обрабатывая события пользователя. Код java-script и DOM способны одни фрагменты документа “на лету” заменять другими или существенно изменять стиль их отображения. Преобразования эти осуществляются на стороне клиента, благодаря чему перезагрузки страницы не требуется. Все выглядит красиво и динамично. Именно благодаря DOM стали возможны все выпадающие меню и списки на страницах браузера. Для корректного формирования объектной модели интерпретатору HTML жизненно необходимо исправлять ошибки верстальщиков, добавляя закрывающие теги, исправляя частичные пересечения области действия тегов разметки (пример выше) и т.п. Обработчик XML, в свою очередь, больше походит на компилятор: он обрабатывается сразу весь документ. Если документ не

является well-formed, то обработчик (парсер) сообщает об ошибке и отменяет формирование DOM целиком. Логично, что в случае XHTML происходит все то же самое. Для XML и его расширений действует правило: “Либо все, либо ничего”, в то время как парсер HTML не сообщает об ошибках, старается их исправлять и практически всегда производит, как минимум, частичную обработку HTML страницы. Кому-то такое поведение HTML интерпретатора может нравиться больше, чем категоричность XHTML обработчика, но позволю себе сделать следующее замечание. Исправление HTML происходит на стороне клиента, т.е. интернет браузером. Различных браузеров достаточно много, не говоря уже о большом количестве версий каждого из них. Это обстоятельство не гарантирует вам, что во всех случаях ошибки будут исправлены одинаково и результат этого исправления вас устроит. Также можно утверждать, что на исправление ошибок тратится дополнительное время. Не стоит в этом полагаться на браузер.

Как итог всему вышесказанному. На данный момент не все интернет браузеры поддерживают XHTML. Для того чтобы поэкспериментировать с этим новым стандартом локально, достаточно файлу с гипертекстом дать расширение .xhtml и открыть его в поддерживающем XHTML интернет браузере, например, в Opera. Если разметка не будет соответствовать well-formed XML, то браузер выведет сообщение об ошибке. Если у тега не будет определено пространство именxmlns=http://www.w3.org/1999/xhtml, то вы увидите просто xml. Ниже приведен пример XHTML разметки страницы:

```
<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>xhtml sample</title>

</head>

<body>

<h1>It is <b>xhtml</b> parsing result</h1>

</body>

</html>
```

Как заставить браузеры использовать XHTML обработчик для страниц, размещенных на удаленных серверах, вы можете найти в интернете, но

заниматься этим сейчас, с моей точки зрения, не обязательно. При разработке сайтов я рекомендую придерживаться правил XHTML, и проверять страницу на соответствие этому стандарту локально. А какой обработчик будет использован браузером пользователя не так важно, поскольку если страница соответствует синтаксису XHTML, то синтаксису HTML она соответствует точно. Принудительно использовать парсер XHTML можно заставить браузер в любой момент, настроив должным образом заголовки интернет страниц на сервере.

Следующий текст может быть полезен как авторам интернет ресурсов и начинающим веб-мастерам, так и программистам, использующим HTML, как инструментоформления отчетов в процессе трансформации данных из других форматов, в том числе из формата XML посредством XSL. Если формируемая разметка не должна стать частью интернет ресурса общего пользования, то на все рассуждения про оптимизацию страниц с целью продвижения их в поисковой выдаче можно не обращать особого внимания. Материал предлагаю воспринимать, как краткий справочник-учебник по основам языка разметки гипертекста и применения каскадных таблиц стилей.

Следующим образом должен выглядеть шаблон интернет страницы на XHTML:

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Заголовок страницы</title>

<meta http-equiv="Content-Type" content="text/html; charset=windows-1251"/>

<meta name="description" content="Описание страницы"/>

<meta name="keywords" content="Ключевые слова"/>

<link rel="stylesheet" href="Путь к файлу каскадных таблиц стилей .css" type="text/css"/>

</head>

<body>

</body>

</html>

Помимо самой разметки, для поисковых систем в интернете (Google, Яндекс и других) важна *метаинформация* о странице. К метаинформации относят содержимое мета-тегов (<meta>) и тега заголовка страницы (<title>). Если располагать их по степени важности для поискового сервиса, то на первом месте идет содержательность и уникальность (по отношению к другим страницам интернет ресурса) заголовка страницы, на втором месте - более развернутое (символов 150-250) описание страницы (description) и на третьем месте уже список ключевых слов (keywords) через запятую.

В чем их важность?

Во-первых, содержимое заголовка – это подпись-ссылка на вашу страницу в поисковой выдаче, а содержимое тега description – это один из кандидатов на роль сниппета (краткого описания страницы) все в тех же результатах поиска. На самом деле те же Google и Яндекс стараются создать сниппет из основного текста вашей страницы, но это у них не всегда получается, поскольку текста может быть совсем мало или не быть вообще, к примеру, на странице размещены только картинки, видео или анимация. В этом случае им не остается ничего другого, как обратиться к краткому описанию страницы. Имея возможность управлять представлением ссылки на ваш сайт, вы можете сделать ее более привлекательной для пользователей поисковой системы и тем самым повысить ее показатель CTR (click-through rate) – количество переходов, деленное на количество показов страницы в результатах поиска.

Во-вторых, исходя из назначения мета-тегов, они должны содержать именно заголовок, отражающий тему страницы, ее краткое описание и ключевые слова. Следовательно, сопоставив мета-теги и само содержание страницы можно понять, насколько оно (содержание) может соответствовать ожиданиям потенциальных пользователей, когда те увидят заголовок и сниппет на странице результатов поискового запроса и примут решение перейти по ссылке. В информационном поиске это соответствие также называют степенью релевантности или просто релевантностью результатов запроса, и именно релевантность, представленная в виде числовых величин, полученных по тщательно скрываемым от общественности алгоритмам оказывает значительное влияние на позиции того или иного сайта в рейтингах поисковиков.

Мета-теги с атрибутом http-equiv играют роль заголовков HTTP запросов, и содержат различного рода системные настройки и параметры, необходимые браузеру, чтобы корректно отображать содержимое веб-страниц. В данном

примере с помощью такого тега браузеру сообщается, что формат страницы – это текст или гипертекст text/html, для отображения которого следует использовать кодировку windows-1251. В русскоязычном сегменте всемирной паутины в основном используют указанному мной кодировку или “универсальную”, построенную на основе 8-ми битного представления юникода, кодировку UTF-8. Если кодировка в мета-теге указана не будет, а браузер пользователя не сможет определить ее самостоятельно, то текст может предстать в нечитаемом виде.

2 ОСОБЕННОСТИ РАБОТЫ С ЯЗЫКОМ HTML

2.1 Техника разметки текста с помощью HTML

HTML (язык гипертекстовой разметки) — это основной язык, с помощью которого создаются веб-страницы. Страница, которую пользователь видит в своем браузере, может состоять из множества разных файлов — например, изображений, анимационных роликов, сценариев JavaScript, апплетов и т. д. - но основой страницы практически всегда является документ HTML. Другие языки разметки (прежде всего многочисленные приложения XML) пока еще слабо поддерживаются браузерами и поэтому не вытесняют HTML на компьютерах обычных пользователей.

CSS (каскадные таблицы стилей) — это язык, предназначенный для оформления веб-страниц и некоторых других видов документов. Разработчик стандартов HTML и CSS, Консорциум Всемирной Паутины (The World Wide Web Consortium, W3C) рекомендует разделять структуру и представление — т. е. кодировать в HTML только логическую структуру документа, а все, что связано с отображением документа на экране или представлением на других устройствах, выполнять средствами CSS.

Аббревиатура «HTML» расшифровывается как «HyperText Markup Language» (в переводе на русский язык — «язык разметки гипертекста»). Взрывной рост Всемирной Паутины в начале 1990-х во многом обусловлен широким распространением этого языка и браузеров, отображающих написанные на нем страницы.

Все три слова в названии языка — «структурный», «разметка» и «гипертекст» - нуждаются в дополнительных пояснениях, которые сейчас и будут даны.

В коде документа HTML находится не только сама информация, которую пользователь увидит на веб-странице, но и некоторые инструкции о том, как браузер пользователя будет обрабатывать эту информацию.

Например, можно включить в документ такую конструкцию, как заголовок заключенного в тэги H1. Пользователь увидит на странице только сам текст заголовка. Окружающие его тэги H1 не отображаются на странице, но снабжают браузер информацией, позволяющей правильно интерпретировать этот текст. Они обозначают, что такой текст является заголовком первого уровня. Большинство современных графических браузеров по умолчанию выделяют этот заголовок полужирным шрифтом увеличенного кегля. (Забегая вперед, заметим, что многие поисковые системы придают заголовкам больший вес, чем обычному тексту, а заголовкам первого уровня — в свою очередь, больший вес, чем прочим заголовкам.)

Язык HTML предназначен в первую очередь для структурной разметки, т. е. для обозначения каждого элемента в соответствии с его местом и ролью в структуре документа. Например, тэг ** … ** обозначает, что элемент является просто абзацем обычного текста, тэг **
…
** — что элемент является перечнем, тэг **<blockquote>…</blockquote>**— что элемент является цитатой и т. д.

В HTML есть также тэги, управляющие внешним видом отдельных элементов: например, тэг **…** выделяет текст курсивом, тэг **…** изменяет различные параметры шрифта, а тэг **
** вставляет перевод строки. Все такие тэги не относятся к структурной разметке, и применять их не рекомендуется. Все, что касается оформления веб-страницы, ее внешнего вида, можно и нужно делать с помощью CSS.

Структурная разметка не имеет никакого отношения к внешнему виду документа. Документ может быть отображен на экране компьютера, написан от руки или напечатан на пишущей машинке; в конце концов, его можно прочитать вслух — но все заголовки в нем останутся заголовками, все перечни — перечнями, а цитаты — цитатами. Структурная роль элементов документа, в отличие от оформления, никак не меняется в зависимости от того, какими средствами просматривают этот документ.

Иными словами, структурная разметка говорит о том, чем является тот или иной элемент, а не о том, как его следует или не следует отображать. Грамотная структурная разметка обеспечивает независимость документа от устройства

вывода.

Структурная разметка не говорит о том, как конкретно должен быть выделен текст. Но это не мешает разработчику оформить его нужным образом с помощью CSS. Технология CSS позволяет даже написать отдельные таблицы стилей для каждого возможного устройства отображения.

Хорошая структурная разметка также облегчает автоматическую обработку документа. Например, когда необходимо пронумеровать все заголовки или составить оглавление документа — программа легко справится с этой задачей, если заголовки помечены соответствующими структурными тэгами. Можно при этом включить в середину документа новые заголовки — программе не составит труда перенумеровать все заново и обновить оглавление. Если не использовать для заголовков правильные тэги H1, H2 и т. д., то их нумерация или составление оглавления превратится в кропотливую и неэффективную ручную работу.

2.2 Особенности работы с тэгами в валидаторе W3C

Собственно говоря, первая версия языка Html появилась в начале девяностых годов прошлого века (жутковато звучит — прошлого века, однако, это именно так) и была ориентирована в первую очередь на передачу информации в научной среде, но, однако, с этого момента можно начинать отчет популярности всемирной паутины.

До этого интернет был уделом немногих знающих и интересующихся людей, но с появлением ХТМЛ и первых браузеров, способных интерпретировать его незамысловатый код в понятные и удобные пользователю вещи (веб страницу или, как еще часто говорят, веб документ), всемирная паутина начала свое победоносное шествие. Наверное, появление языка гипертекстовой разметки можно приравнять к переходу от текстовых к графическим операционным системам.

Довольно интересными на мой взгляд представляются темп и нюансы развития этого языка разметки. Итак, через несколько лет после появления первой версии, по инициативе Тима Бернерса-Ли был создан консорциум W3C(World Wide Web Consortium), призванный был стать законодателем стандартов и не допустить разброда и шатаний в рядах разработчиков, которые могли привести к очень неприятным последствиям.

В 1994 году разрабатываются стандарты гипертекстовой разметки второй версии, а уже в 1995 ведутся работы над Html 3 с поддержкой CSS (таблиц каскадных стилей). Примерно в это же время появляется и набирает популярность первый браузер Мозаика, который очень быстро был переделан в Netscape Navigator.

MicroSoft хотела купить Netscape Navigator для интеграции его в Windows, но разработчики этого браузера отказались (в итоге появился Mozilla Firefox), в результате чего мы получили и имеем по сей день собственное творение MicroSoft (IE — Internet Explorer), которое они создали на базе открытых кодов Мозаики.

Что интересно, IE (в силу своей предустановленности в самой популярной операционной системе) сумел-таки выкинуть из рейтинга браузеров некогда очень популярный Netscape Navigator, но получил взамен ряд новых игроков (бесплатный браузер Opera, обозреватель от Mozilla, а еще сегодняшнего лидера Гугол Хром и др.). В этот период разработчики браузеров зачастую бежали впереди паровоза (валидатора) и вводили свои собственные стандарты, ибо работа над форматом Html в W3C шла довольно медленно.

Узрев такое дело, консорциум в течении одного 1997 года сделал огромный скачок — язык разметки претерпел сразу два изменения, перейдя от версии 3.2 до версии 4.0, а затем (в 1999) и до той версии, которую мы используем по сей день — **Html 4.01**. С тех пор, представляете, стандарт не менялся уже около двенадцати лет (всех все устраивало, и разработчиков браузеров и команду валидатора W3C).

Сейчас консорциум с подачи конгломерата разработчиков браузеров ведет активные работы над форматом, но ждать его появления в ближайшем будущем, наверное, не стоит. Хотя некоторые нововведения **формата ХТМЛ 5**уже поддерживаются некоторыми браузерами в той или иной степени. Скорее всего, именно так и будет происходить дальше — новый формат будет внедряться по частям, но полной его поддержки всеми браузерами придется ждать довольно долго.

Множество web страниц, относящихся к одному и тому же доменному имени, называются сайтом. Понятно, что web страницы, входящие в состав сайта, должны где-то физически размещаться и быть доступными любым пользователям в течении 24 часов.

Для этих целей предназначены web сервера, которые, как правило, размещены в специально оборудованных для этого помещениях, и представляют из себя обычные компьютеры, заточенные и оптимизированные под выполнение данной

задачи. Web сервера имеют постоянный и широкополосный доступ в интернет для того, чтобы пользователи из любой точки земного шара и в любом количестве могли получить доступ к вашему сайту.

Услуги по предоставлению места для размещения сайтов предлагают так называемые хостеры и, естественно, делают они это за деньги, хотя в природе встречаются и приятные исключения в виде бесплатного хостинга.

Любая страница (документ) сайта будет иметь свой собственный уникальный адрес или, другими словами, URL (Uniform Resource Locator). Для того, чтобы пользователь увидел содержимое этой страницы у себя на экране компьютера, его браузер должен будет завязать диалог с сервером путем посылки http запросов и получения http ответов.

В результате этого диалога браузер получает Html код документа, разбирает его, подгружает все необходимые дополнительные элементы оформления страницы (изображения, ccs файлы, скрипты) и пользователь видит на экране страницу сайта.

Html тэги для начинающих и их список в валидаторе

Любой документ будет состоять из двух частей:

1. То содержимое, которое мы хотим вывести в том или ином виде на web странице, т.е. то, что мы размечаем средствами языка гипертекстовой разметки
2. **Html тэги** (само средство разметки). Их можно отличить от содержимого страницы по угловым скобкам, в которые они будут обязательно заключены (<тэг>). Иногда их еще называют дескрипторами, но чаще все же употребляется именно термин «теги».

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>KtoNaNovenkogo.ru - все для начинающих вебмастеров</title>
<link rel="alternate" type="application/rss+xml" title="RSS 2.0" href="http://ktonanovenkogo.ru/feed" />
</head>
<body>
</body>
</html>
```

Править или писать Html теги лучше всего будет в специализированном редакторе, таком, например, как Notepad++ с подсветкой синтаксиса.

Сами теги опять же можно разделить на два простых вида:

1. Открывающий, после которого размещается то содержимое страницы, которое мы хотим отформатировать или структурировать с помощью него. Пример такого тэга может выглядеть так:
   <body >.
2. Закрывающий, который выглядит так же как и открывающий, но сразу после первой угловой скобки в закрывающем тэге прописывается прямой слеш (</body >) перед его названием

Вот наглядный пример закрывающего и открывающего Html тега, который позволяет разметить заключенный в него текст, как заголовок третьего уровня:

<h3> Заголовок статьи </h3>

Да, еще существуют так называемые одиночные или же, по-другому, пустые тэги, которые попросту не имеют закрывающего элемента и таким образом в них нельзя что-либо заключить для разметки, но зато с помощью них можно будет что-либо вывести на страницу (например, изображение).

В языке разметки ХТМЛ имеется четко ограниченный и заранее составленный набор тегов, которые подробнейшим образом описаны на соответствующей стандарту странице валидатора W3C. Само собой разумеется, что лучше всего будет пользоваться именно этим списком разрешенных элементов, ибо это **первоисточник** и не допускает двойственных толкований.

Для этого на странице валидатора W3C со списком стандартов XTMЛ, которые относятся к различным спецификациям, вам нужно будет перейти по ссылке «HTML 4.01 Specification».

На открывшейся странице нужно перейти на вкладку «elements» из верхнего меню, после чего вы сможете, наконец-то, лицезреть список Html тегов **валидных на данный момент** по версии спецификации 4.01, разработанной аж в далеком 1999 году (в прошлом веке).

Name	Start Tag	End Tag	Empty	Depr.	DTD	Description
A						anchor
ABBR						abbreviated form (e.g., WWW, HTTP, etc.)
ACRONYM						
ADDRESS						information on author
APPLET				D	L	Java applet
AREA		F	E			client-side image map area
B						bold text style
BASE		F	E			document base URI
BASEFONT		F	E	D	L	base font size
BDO						I18N BiDi over-ride
BIG						large text style

В столбце «Description» списка вы найдете краткое описание интересующего вас тэга, а прочитать подробнейшее описание сможете, щелкнув по его названию. В колонке «Start Tag» (начальный элемент) напротив некоторых из них можете наблюдать проставленную букву «O», означающую, что данный открывающий тег можно использовать опционально, т.е. можно его использовать, а можно и не использовать.

Таких элементов в списке всего четыре, и три из них задают общую структуру всего документа (<html>, <body> и <head> — о них мы отдельно поговорим чуть ниже). В столбце списка под названием «End Tag» (закрывающий элемент) вы тоже можете встретить пометку «O» (опционально).

Что это означает? Дело в том, что стандарт гипертекстовой разметки **4.01 делает нам послабление** и (зачем-то) разрешает не писать помеченные буквой «O» закрывающие или открывающие элементы. За нас это сможет сделать браузер, разбирая полученный от сервера Html код.

Т.е. касаемо структуры документа, о которой мы поговорим чуть ниже, браузер сам сможет создать необходимые ему разделы <html>, <body> и <head>, даже если вы забудете это сделать. Кроме этого, вы можете ставить, например, только

открывающий тэг параграфа (абзаца) , а закрывающий можно игнорировать, т.к. в любом случае за вас его проставит браузер при разборе кода разметки.

Еще одним послаблением валидатора W3C в спецификации 4.01 является то, что тэги можете писать в любом удобном вам регистре (хоть большими буквами, хоть маленькими, а хоть и вперемешку). Т.е. язык **ХТМЛ является регистронезависимым**.

Это, конечно же, очень здорово, что нам так много позволено, но дело в том, что в следующей версии Html 5, похоже от всех этих двусмысленностей попытаются отказаться. В результате чего придется писать все открывающие и закрывающие элементы, а также сам язык будет регистрозависимым. Поэтому, наверное, имеет смысл уже сейчас не начинать привыкать к этим вольностям и закрывать все тэги, а еще писать их названия в нижнем регистре.

На странице валидатора, где приведен список тегов стандарта 4.01, имеется еще один столбец с названием «Depr», где буквами «D» помечены **не рекомендованные к использованию элементы**. Если посмотрите в списке что это за теги, которые не рекомендуются к применению, то увидите, что они в основном отвечают за оформление содержимого документа (например, FONT или).

Дело в том, что сейчас для визуального оформления Html документов принято использовать CSS (таблицы каскадных стилей), которые имеют гораздо больше возможностей, нежели не рекомендованные к использованию в W3C элементы. Хотя такое к ним отношение в validator вовсе не означает, что про них вы можете забыть.

Эти не рекомендованные тэги можете встретить в ХТМЛ документах интернета, а кроме этого, в случаях, когда нельзя использовать оформление через CSS (все та же почтовая рассылка Subscribe), то эти самые не рекомендованные элементы могут вам очень сильно помочь и пригодиться.

ЗАКЛЮЧЕНИЕ

Становится все более очевидным, что язык этот в его теперешнем состоянии практически исчерпал перспективы развития, и что добавление новых тегов вряд

ли выведет его на принципиально иной уровень.

Это особенно понятно тем, кто занимается практическим дизайном: из-за того, что HTML с самого начала не ориентировался на описание внешнего вида документа, он не в состоянии удовлетворительно выполнить эту задачу даже сейчас, при наличии множества визуально-ориентированных тегов. Прямым следствием этого является огромное количество расхождений в интерпретации тегов браузерами.

Как бы строго вы ни следовали стандарту, HTML-файл приходится обязательно тестировать по меньшей мере в двух броузерах - NetscapeNavigator и InternetExplorer, и чаще всего такое тестирование не обходится без неприятных сюрпризов: отступы, пробелы, размеры элементов оформления и логика их размещения на странице даже для простейших тегов различаются довольно сильно.

СПИСОК ЛИТЕРАТУРЫ

1. Биячуев, Т. А. Безопасность корпоративных систем: учеб. пособие / Т. А. Биячуев. – СПб.: СПб ГУ ИТМО, 2014. – 161 с.
2. Браун, С. Виртуальные частные сети : учеб. пособие / С. Браун. – М: Лори, 2015. – 481 с.
3. Романец, Ю. В. Защита информации в компьютерных системах и сетях: Учебное пособие / Ю.В.Романец, П.А.Тимофеев, В.Ф.Шаньгин. – М. : Ифра-М, 2015. – 304 с.
4. Дворский, М. Н. Техническая безопасность объектов предпринимательства : Учебное пособие / М. Н. Дворский, С. Н. Палатченко. – М. : А-депт, 2016. – 304 с.
5. ViPNet Администратор: Руководство администратора
6. ViPNet Координатор: Руководство администратора
7. http://www.infotecs.ru
8. http://expo-infosecurity.ru
9. http://ru.wikipedia.org/wiki/VPN