XHTML

УРОКИ ДЛЯ ПОЛУСТАЦИОНАРА

УРОК №1

ВВЕДЕНИЕ В WEB-ТЕХНОЛОГИИ. ФОРМАТИРОВАНИЕ ТЕКСТА.

ОГЛАВЛЕНИЕ

1. Введение в языки разметки. Спецификации языка HTML

Итак, мы начинаем изучение курса основ web-дизайна или XHTML и CSS, который познакомит вас с основами разметки текста с помощью современной технологии XHTML. Так же мы изучим возможности визуального оформления элементов web-страниц с помощью каскадных стилевых таблиц - CSS.

XHTML — это новая версия очень популярного и широко распространенного языка гипертекстовой разметки (HTML — Hypertext Markup Language), построенная в соответствии с правилами расширяемого языка разметки (XML — Extensible Markup Language). Фактически, XHTML наследует простоту, а также синтаксис и разметку HTML и заставляет его функционировать по правилам XML при описании и обработке разметки.

История развития XHTML

SGML

Начало истории HTML следует отнести к далекому 1969 году, когда Чарльз Гольдфарб, работавший тогда в компании IBM, создал прототип языка для разметки технической документации, впоследствии названного GML, а с приданием ему в 1986 году статуса международного стандарта — SGML (Standard Generalized Markup Language). Этот обобщенный метаязык предназначен для построения систем логической, структурной разметки любых разновидностей текстов. Слово «структурная» означает, что управляющие коды, вносимые в текст при такой разметке, не несут никакой информации о форматировании документа, а лишь указывают границы и соподчинение его составных частей, т.е. задают его структуру.

Создатели SGML стремились полностью абстрагироваться от проблем представления текста в разных программах, на разных компьютерных платформах и устройствах вывода. Хотя формально ничто не мешает записать средствами SGML любую информацию об элементах документа - в том числе и параметры его форматирования (к примеру, шрифр, размер, цвет текста заголовка), идеология этого языка требует ограничиться указанием на уровень заголовка и его место в иерархической структуре документа. Все остальное должно быть вынесено в так называемые стилевые спецификации.

Сам по себе SGML есть не готовая система разметки текста, а лишь удобный метаязык, позволяющий

строить такие системы для конкретных обстоятельств.

HTML 1.2

Принципы, на которых строится язык SGML, значительны и интересны; несомненно, идеология языка оказала влияние на многие компьютерные разработки. Однако сам по себе SGML не получил сколько-нибудь заметного распространения до тех пор, пока в 1991 г. сотрудники Европейского института физики частиц (CERN), занятые созданием системы передачи гипертекстовой информации через Интернет, не выбрали SGML в качестве основы для нового языка разметки гипертекстовых документов. Этот язык - самое известное из приложений SGML - был назван HTML (HyperText Markup Language, "язык разметки гипертекста").

Первым (и единственным в те далекие времена) графическим броузером была программа Mosaic, разработанная, как и сам WWW, в научном учреждении - Национальном Центре Суперкомпьютерных Приложений США (NCSA). Так что нет ничего удивительного в том, что в этот "золотой век" никаких противоречий между официальными стандартами и их реализацией в броузерах еще не существовало.

HTML 2.0

HTML неторопливо развивался, оставаясь в рамках парадигмы структурной разметки, и в апреле 1994 г. началась подготовка спецификации следующей версии языка — 2.0. Этим занимался образованный в том же году Консорциум W3 (W3 Consortium, сокращенно W3C), перенявший от CERN верховную власть и авторитет в мире WWW.

В настоящий момент консорциум, имеющий статус «международного и некоммерческого», объединяет свыше 150 организаций-членов, в том числе фирмы Netscape, Microsoft и множество других. Однако в 1994-1995 гг. его членами были почти исключительно университеты и научные учреждения. Столь академический состав W3C сказывался как на самих документах, публикуемых консорциумом, так и на процедуре (и особенно на сроках) их принятия. Достаточно сказать, что спецификация HTML 2.0, единственным серьезным усовершенствованием в которой был механизм форм для отсылки информации с компьютера пользователя на сервер, была окончательно утверждена лишь в сентябре 1995 г., когда в W3C уже полным ходом шло обсуждение HTML 3, - или, как его называли поначалу, "HTML+".

HTML 3

Пожалуй, проект HTML 3 — самая яркая и неоднозначная страница в истории языка. Работа над ним началась в марте 1995 г., и первоначальный вариант стандарта включал в себя много интересных нововведений. Но самое главное — HTML 3 был попыткой разрешить уже достаточно очевидное к тому времени противоречие между идеологией структурной разметки и потребностями пользователей, заинтересованных в первую очередь в гибких и богатых возможностях визуального представления.

Противоречие это было разрешено опять-таки в полном соответствии с идеологией SGML: W3C ввел в HTML 3 поддержку так называемых иерархических стилевых спецификаций - CSS. Система CSS формально независима от HTML, имеет совершенно иной синтаксис, не наследует никаких идеологических ограничений и позволяет, уже в совершенно иных терминах, задавать параметры графического представления для любого тега HTML. Нет сомнения, что CSS — почти идеальный способ избавить HTML от наследственных дефектов и перевести его развитие на принципиально новые рельсы.

“ВОЙНА БРАУЗЕРОВ”

Коммерческое освоение WWW не заставило себя долго ждать. В начале 1994 г. группа разработчиков броузера Mosaic основала корпорацию Netscape Communications и вскоре выпустила первую версию коммерческого броузера Netscape (начиная с версии 2.0 — Netscape Navigator, а с версии 4.0 — Netscape Communicator). С этого момента начался экспоненциальный рост WWW, продолжающийся по сей день. Чтобы закрепить лидерство (на которое, впрочем, тогда еще мало кто покушался) и привлечь новых пользователей, Netscape вводила в HTML все новые и новые усовершенствования, — поддерживаемые, разумеется, только броузером Netscape.

Практически все новые теги, без устали изобретаемые Netscape, были направлены на улучшение внешнего вида документа и расширение возможностей его форматирования. Причины понятны: чтобы убедить, скажем, бизнесмена, что ему пора обратить внимание на некую новую технологию, прежде всего нужно показать ему ее в привлекательном, «товарном» виде. В результате тот вариант HTML, который поддерживала выпущенная в начале 1996 г. версия Netscape Navigator 2.0, представлял собой довольно странную смесь старых логических тегов с беззастенчиво вломившимися новыми, ориентированными на графическое экранное представление документа и затрудняющими его воспроизведение на других устройствах вывода.

Такая политика компании, с одной стороны, принесла ей быстрый и впечатляющий успех (одно время версии Netscape Navigator составляли более 90% всех используемых броузеров), а с другой — вызвала ожесточенное сопротивление наиболее сознательной части HTML-сообщества. Энтузиасты неустанно разъясняли и разъясняют каждому, кто согласен их слушать, что HTML по природе своей не имеет права зависеть от какого-то конкретного броузера и что заявления типа «эту страницу лучше всего смотреть в Netscape Navigator» являются просто насмешкой над здравым смыслом.

В конце 1995 г. ситуация в мире HTML была довольно смутной. Популярность броузера Netscape неуклонно росла; программисты этой фирмы готовили к выпуску версию 2.0, которая должна была утвердить господство Netscape на вечные времена благодаря неслыханному набору новшеств (интерфейс подключаемых модулей - plugins, поддержка Java-апплетов, встроенный язык сценариев JavaScript, возможность разбивки окна на фреймы и многое другое). В этот переломный момент в игру вступил новый участник — корпорация Microsoft. Долгое время эта компания, привыкшая монопольно владеть своим сектором рынка, недооценивала перспективы Интернета и не собиралась как-либо участвовать в развитии этой информационной среды. Однако невероятный взлет Netscape заставил Microsoft изменить свое мнение.

И именно на броузерном фронте, где господство Netscape оставляло меньше всего шансов конкурентам, корпорация Microsoft нанесла свой главный удар. Поначалу мало кто верил, что броузер Microsoft Internet Explorer, который тогда существовал в версии 2.0 и не представлял собой ничего выдающегося, сможет составить конкуренцию Netscape; Тем не менее выпущенная летом 1996 г. версия Internet Explorer 3.0, которая поддерживала почти все расширения Netscape, вызвала настоящий бум и очень быстро утвер¬дилась в качестве «второго главного броузера». Сейчас Microsoft и Netscape делят рынок броузеров почти поровну, и окончательный исход их битвы не берется предсказать никто.

HTML 3.2 И 4

Одновременно с разработкой конкурентоспособного броузера Microsoft решила “навести порядок” и в мире HTML. Взяв под свою опеку W3C, она снабдила его денежными и людскими ресурсами и тем самым заработала право едва ли не решающего голоса в этой организации. Проект HTML 3 был заморожен, а вместо него в сжатые сроки создан стандарт HTML 3.2, который, по сути, всего лишь описывает большинство расширений Netscape (с тем же успехом их можно назвать теперь “расширениями Microsoft”). Пройдя обычный в W3C процесс

обсуждения и внесения поправок, спецификация HTML 3.2 была утверждена в январе 1997 года. Расхождения между предписаниями стандарта и реализацией HTML в браузерах вновь были сведены к минимуму.

В декабре того же 1997 г., с принятием стандарта HTML 4.0, маятник, похоже, качнулся уже в обратную сторону - наряду с дальнейшим обогащением репертуара визуальных тегов, эта версия ввела немало пусть и не вполне "логических", но очень важных расширений для поддержки многоязычных документов и обеспечения доступности документа в разных средах. Кроме того, в HTML 4 наконец-то прямо в тексте стандарта четко проведено разделение логических и визуальных тегов (последние объявлены "нерекомендованными", "deprecated").

HTML 5

Разработка HTML 5 началась в 2007 году. Первый черновик спецификации стал доступен широкой общественности 22 января 2008. Спецификация сайчас находится в разработке, и может находится в ней несколько лет, хотя отдельные части HTML 5 будут закончены и реализованы в браузерах до того, как спецификация официально получит статус рекомендации.

HTML 5 вводит несколько новых элементов и аттрибутов. Это будет облегчать работу поисковикам, а также обработку сайта с КПК или читающих программ. Другие элементы совершенно новы и представляют новую функциональность. Некоторые устаревшие элементы HTML 4 были удалены из HTML 5.

В отличие от четвёртой версии, пятая версия чётко прописывает правила лексического разбора, чтобы различные браузеры отображали один и тот же результат в случае некорректного синтаксиса.

XML

XML - набор правил и соглашений о синтаксисе, с помощью которого можно создавать собственные наборы элементов разметки. Эти элементы, в свою очередь, можно использовать для описания содержимого. Своим появлением XML обязан невозможности применения HTML для описания данных самого разного рода, которые пользователи желают распространять через Web. Так, HTML практически невозможно использовать для описания финансовых данных, руководств по инсталляции программного обеспечения, математических выражений и множества данных других типов, которыми буквально наполнена сеть Web.

Хотя изначально HTML разрабатывался как язык разметки (описывающий содержимое и не зависящий от внешнего вида), со временем он превратился в язык дескрипторов, используемый разработчиками для настройки внешнего вида и передачи смысла информации Web-страницы. Распространение данных, тем не менее, характеризуется рядом особенностей, которые не вполне успешно учитываются в HTML.

Для достижения гибкости и согласованности, отсутствующей в HTML, и был разработан XML. Этот язык обеспечивает возможность создания собственных наборов элементов разметки, что позволяет при представлении содержимого выйти за пределы ограничений стандартных наборов HTML. Истинное назначение языка разметки - описывать части документа, не касаясь способов его визуализации и отображения. Разметка позволяет структурировать данные, а уже затем эти данные можно использовать по-разному.

XHTML

Язык XHML представляет собой словарь XML. Он построен в соответствии с правилами спецификации XML 1.0, поэтому документы XHTML являются одновременно и документами XML. XHTML - это измененная версия HTML 4.01, переписанная в соответствии с правилами XML.

На сегодняшний день существует несколько спецификаций XHTML:

XHTML 1.0 Transitional - является переходным вариантом от традиционного HTML к XHTML. Требования к структуре документа и синтаксис не столь строгие как в XHTML

XHTML 1.0 Strict - спецификация для описания обыкновенного XHTML документа, обладающая всей строгостью XML.

XHTML 1.0 Frameset - спецификация XHTML для описания документа с фреймовой структурой (frameset).

XHTML 1.1 - в данной спецификации получили воплощение принципы модульной разметки. Модульный подход к развитию XHTML является ключевым элементом, благодаря которому отдельные пользователи и целые организации смогут применять для своих целей спецификацию XHTML, не тратя время на разработку частной разметки или версий спецификации. Модуляризация XHTML обеспечивает формальный механизм расширения XHTML в существующих рамках синтаксиса и соглашений XML.

XHTML 2.0

XHTML 2.0 — язык разметки, разработанный с целью предоставить документы для широкого диапазона целей Всемирной Паутины (World Wide Web). Спецификация на данный момент находится в разработке.

Многие пункты в спецификации XHTML 2.0 проекта являются спорными, потому что они нарушают обратную совместимость со всеми предыдущими версиями, и поэтому, в действительности, XHTML 2.0 — это новый язык разметки, созданный специально, чтобы обойти ограничения (X) HTML, а не просто стать новой версией.

За дополнительной информацией по каждой из вышеперечисленных спецификаций посетите сайт W3C:

http://www.w3.org

2. Краткий обзор броузеров и их возможностей.

Именно просмотр web-страниц и является основной функцией браузера, и то, как он с ней справляется, определяет его позицию в рейтинге популярности.

Помимо основной функции, ряд браузеров поддерживает дополнительные возможности, позволяющие упростить и сделать более комфортной работу пользоватлей в мировой сети Internet. Среди этих возможностей можно особо выделить поддержку просмотра множества страниц в одном окне (на вкладках), блокирование нежелательной рекламы (банеров и всплывающих окон), быстрый доступ к различного рода опциям загрузки страницы (отключение изображений, анимации и пр. для умеличения скорости загрузки).

Ниже приводится перечень наиболее популярных на сегодняшний день браузеров:

- Microsoft Internet Explorer ;
- Mozilla Firefox ;
- Opera ;
- Safari;
- Google Chrome.

Безусловно, это далеко не полный перечень, однако, мы не будем заострять внимания на остальных браузерах, не вошедших в этот список.

Microsoft Internet Explorer (MSIE)

Данный браузер, безусловно, является лидером в современном первенстве браузеров. Созданный Microsoft Corporation, он приобрел широкую известность, в основном благодаря авторитету и политике компании.
MSIE давно поставляется как часть операционной системы Microsoft Windows, и поэтому доступен любому ее пользователю.

Mozilla Firefox

Это совсем молодой браузер, созданный компанией разработчиков браузера Mozilla - прародителя большинства совеменных браузеров. Год “рождения” Firefox - 2004, так что он по праву может считаться наиболее современным. Из числа наиболее достопримечательных дополнительных возможностей можно выделить:

– поддержка вкладок, позволяющих просматривать множество страниц в одном окне программы, не занимая много места на панели задач;

– быстрый доступ к таким функчиям, как отключение загрузки изображений (экономи траффика и ускорение загрузки документа);

– поддержка альтернативнх схем оформления внешнего вида (skins) и дополнительных модулей (plugins), расширяющих его возможности;

– просмотр и анализ исходного кода страницы на предмет соответствия современным стандартам (только после установки дополненительного модуля).

Кроме всего вышеперечисленного, Firefox имеет встроенный менеджер закачек, с поддержкой докачки файла при обрыве соединения.

Opera

Приятный и удобный интерфейс Opera легко настраивается. Браузер имеет встроенный менеджер закачек с поддержкой докачки файла при обрыве соединения и встроенный почтовый клиент. Просмотр страниц осуществляется на вкладках, что экономит место на панели задач Windows. Для пользователей, требовательных к внешнему виду программ, приятной особенностью будет поддержка схем оформления внешнего вида - skins.

Safari

Safari — веб-браузер разработан компанией Apple, входит в состав операционной системы Mac OS X, а также бесплатно распространяется для ОС Windows. Данный веб-браузер создавался когда подходил к концу срок действия договора Apple с Microsoft о поддержке Internet Explorer для платформы Macintosh. Вскоре после появле

ния Safari работа над IE для Macintosh была прекращена.

Версия для Windows обладают такой же функциональностью, как и под Mac OS X (включая Bonjour). Внешний вид программы отличается от обычного вида программ в Windows: элементы интерфейса (кнопки прокрутки, поля ввода, выпадающие меню, чекбоксы и др.) выполнены в стиле Mac OS X.

Основные возможности: использование вкладок, удобный и простой поиск фрагмента текста на странице, частный просмотр, масштабирование области ввода текста, автоматически распознаёт веб-сайты, использующие нестандартные шрифты, и загружает их по мере необходимости, позволяет пользователям и разработчикам просматривать Document Object Model (DOM) веб-страниц.

Google Chrome

Google Chrome — это браузер, обеспечивающий более удобную, быструю и безопасную работу в Интернете, который имеет упрощенный дизайн, облегчающий работу.

Функции:

- одно окно для любых задач;
- ярлыки приложений;
- динамические вкладки;
- контроль сбоев;
- режим инкогнито;
- безопасный просмотр;
- мгновенное создание закладок;
- импорт настроек;
- упрощенный процесс загрузки.

Резюме

При разработке страниц вам необходимо пользоваться несколькими браузерами, при этом вы научитесь разрабатывать оптимальные страницы под браузеры различных производителей.

3. Краткий обзор редакторов и их возможностей.

Прежде чем приступать к изучению структуры XHTML документа и набора элементов и правил их записи, необходимо, так же, определиться с выбором инструментов для работы с документом. Основным инструментом, используемым для создания web-страниц, является текстовый редактор. Для этой цели подойдет практически любой редактор (к примеру, "Блокнот" Windows). В профессиональной же деятельности web-мастера, к которой, несомненно, готовятся некоторые из вас, огромное значение играет не только возможность правки текста, а еще и ряд дополнительных характеристик редактора, облегчающих процесс разработки и расширяющих его функциональность:

- **Поддержка различных кодировок текста.** Рекомендуемой кодировкой текста для документов XHTML является "UTF-8", однако, web-страница может быть создана и с использованием другой кодировки (например, KOI8-R или Windows-1251). Таким образом, чем шире спектр поддерживаемых редактором кодировок текста, тем меньше вероятность попасть в тупиковую ситуацию - не имея возможности отредактировать документ, созданный в неизвестной кодировке. Более подробно кодировки будут рассматриваться нами в одном из следующих уроков.

- **Подсветка синтаксиса.** Создавая текстовые XHTML документы высокой сложности, вы скоро придете к выводу, что использование редактора, не делающего различия между элементами разметки и текстовым содержимым страницы (например, "Блокнот" Windows) сильно затрудняет визуальный разбор документа и его правку. Существует ряд редакторов, таких как **AceHTML** или **MS Visual Studio**, позволяющих выделить элементы разметки XHTML альтернативным цветом и/или начертанием, и тем самым облегчить разработку и отладку web-страниц.

- **Возможность запуска встроенного или внешнего браузера для просмотра страницы.** Безусловно, любой XHTML документ можно запустить в браузере непосредственно оттуда, где он был сохранен или размещен разработчиком, однако эта процедура требует ряда дополнительных операций, отвлекающих от основной задачи - разработки кода и контента страницы. Знакомые с основами разработки приложений на C++ в среде MS VisualStudio, вы, несомненно, сочтете удобной такую возможность, как запуск документа "на выполнение" нажатием клавиши на клавиатуре, или кнопки на панели инструментов редактора. К примеру, редактор **AceHTML** при установке сам "найдет" установленные в системе браузеры (IE, Opera, Mozilla и пр.) и свяжет с каждым из них сочетание клавиш для быстрого запуска.

- **Поддержка шаблонов кода.** Использование шаблонов кода позволяет избавить разработчика

страниц от не слишком то увлекательного процесса повторного ввода или копирования часто повторяющегося и, к тому же, довольно большого фрагмента страницы. Сохранив один раз такой фрагмент и назначив ему быстрый доступ в виде сочетания клавиш или кнопки на панели инструментов, вы можете в дальнейшем легко вставить его в нужное место на странице. Некоторые редакторы имееют предопределенные разработчиками шаблоны кода.

• **Поддержка функции "Code Completion".** Суть данной фкнкции состоит в подсказке разработчику возможных вариантов продолжения начала вводимой им конструкции. При выборе варианта продолжения, редактор сам "вбивает" недостающие части конструкции, упрощая и ускоряя процесс разработки. Такая функция, к примеру, реализованна в системе разработки **MS VisualStudio.NET**. Редактор подсказывает разработчику возможные варианты продолжения вводимой конструкции не просто так, а в зависимости от контекста, определяя допустимость данной конструкции в данном месте кода страницы.

• **Проверка документа на соответствие стандарту.** Ряд редакторов позволяет выполнять проверку исходного кода страницы на предмет соответствия выбранной спецификации. Процесс разработки страниц в таком редакторе начинает напоминать процесс разработки приложений на современных языках программирования: набор кода, проверка на наличие ошибок и отладка, тестовый запуск.

Далее представлен ряд редакторов, рекомендуемых для использования и их основные возможности.

VisicomMedia AceHTML

Этот редактор примечателен тем, что при широком спектре поддерживаемых возможностей. В числе основных особенностей данного редактора есть:

- поддержка всевозможных кодировок текста;

- качественная и легко настраиваемая подсветка синтаксиса;

- функция Code Completion (работающая, впрочем, не всегда корректно);

- поддержка предопределенных и легко настраиваемых пользовательских шаблонов кода;

- просмотр страниц внутренним или внешним браузером (можно использовать любой установленный в системе браузер, например Opera или Firefox);

- настройка атрибутов и стиля элементов с помощью специальной панели Code Inspector.

MS VisualStudio.NET

Это один из наиболее мощных рассматриваемых нами инструментов разработки. В нем реализованна наиболее мощная поддержка функции Code Completion, грамотная и легко настраиваемая подсветка синтаксиса и работа с текстом в различных кодировках.

К числу его недостатков можно отнести лишь 1.5 ГБ требуемого для установки пространства на жестком диске, высокие требования к производительности компьютера и отсутствие возможности запуска страницы на просмотр в браузере (как внутреннем, так и внешнем).

CSE HTML Validator

Основное достоинство данного редактора - возможность проверки документа на соответствие выбранной спецификации. Так же поддерживаются различные кодировки текста, предопределенные и пользовательские шаблоны кода (последние, впрочем, довольно сложно настроить) и запуск страницы на просмотр во внешнем браузере (для этого используется системный браузер по умолчанию).

Кроме того, данный редактор обладает довольно простенькой, однако, вполне достаточной, подсветкой синтаксиса. Поддержка функции Code Completion отсутствует. Занимает данный редактор не более 10 МБ на жестком диске.

Мы рекомендуем

Для изучения XHTML я не рекомендую использовать автоматические среды для разработки, которые дописывают теги, аттрибуты, свойства и т.д. Лучшим вариантом будет использование программ которые только подсвечивают синтаксис и расставляют табуляцию, что делает код более читаемым(Например: NotePad++). В крайнем случае можно использовать “Блокнот” Windows.

4. Структура HTML-документа.

Итак, после выбора средств разработки, а именно редактора (для создания страниц) и браузера (для их тестирования), мы с вами приступаем непосредственно к рассмотрению структуры документа XHTML.

Первое, что вы должны узнать об XHTML документах, это то, что они являются обыкновенными текстовыми файлами, содержащими специальные текстовые метки, называемые элементами разметки. Набор этих меток составляет структуру XHTML документа и, в общем то, определяет его внешний вид. Набор символов по умолчанию для XHTML документов - UTF-8, однако, ничто не помешает нам воспользоваться и другой кодировкой (о кодировках и их указании речь пойдет в одном из следующих занятий).

Чтобы создать XHTML документ с помощью программы AceHTML 6, воспользуйтесь командой File \ New.... В списке доступных шаблонов выберите XHTML Document. После нажатия кнопки "Ok" вашему вниманию предстанет примерно такая структура:

Листинг №1.4.1: Простой XHTML документ

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Simple XHTMl Document</title>
  </head>
  <body>
    <h1>Простой документ</h1>
    <p>
```

```
        Документ, который вы видите, представляет собой
        простой XHTML документ, содержащий один заголовок и
        один абзац текста...
      </p>
   </body>
</html>
```

пример находится в папке: samples/example_1.4.1.html

Если "выбросить" из приведенного документа все, что располагается между **<body>** и **</body>**, то, что останется, и будет "скелетом" любого XHTML документа.

На данном этапе изучения материала рекомендуется принять как должное такую структуру документа и использовать ее, не вникая глубоко в назначение ряда ее элементов. Некоторые из них рассматриваются ниже подробно, остальные будут рассмотренны по мере необходимости далее.

XML декларация

```
<?xml version="1.0" encoding="utf-8"?>
```

Этот элемент присутствует в любом XML документе, каким так же является и XHTML документ. Декларация XML указывает браузеру на то, что данный документ построен в соответствии с синтаксисом XML разметки версии **1.0** и использует набор символов **utf-8**. В общем случае, xml декларацию можно не указывать, так как кодировкой символов по умолчанию для документов XHTML является именно utf-8. Однако, при использовании альтернативной кодировки (например **windows-1251**), указывать xml декларацию необходимо.

Определение Типа Документа (DTD)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
 "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

DTD, или Определение Типа Документа (Document Type Definition), является еще одной неотъемлимой частью документов XML. Объявление **<!DOCTYPE...>** указывает браузеру, какой набор элементов разметки может быть использован в данном документе.

Созданный в примере выше XHTML документ соответствует спецификации XHTML 1.1.

Корневой элемент - **<html>**

В соответствии с синтаксисом XML, любой документ должет иметь единственный элемент верхнего уровня, в который должны быть вложенны остальные элементы. Для XHTML документов роль корневого играет элемент <html> ... </html>. В него вкладываются остальные элементы, такие как <head> - раздел заголовка и <body> - тело, содержательная часть документа.

Раздел заголовка - **<head>**

В разделе заголовка обычно размещают информацию, характеризующую страницу в целом. Как то:

- Создание заголовка страницы;
- Указание кодировки страницы;
- Перечисление ключевых слов, помогающих поисковым системам проиндексировать вашу страницу;
- Описание или подключение таблиц стилей и сценариев.

Заголовок страницы можно указать при помощи элемента **<title>**. Например так:

```
<head>
  <title>Текст заголовка страницы</title>
  ...
</head>
```

Этот текст используется различными браузерами по разному. MSIE отображает его в заголовке окна браузера, а Firefox и Opera - в виде названия вкладки, на которой загружена страницы:

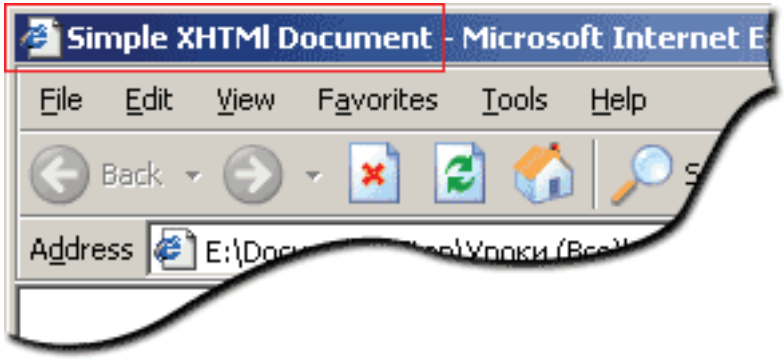

Рис. №1.4.1. Заголовок страницы. MSIE

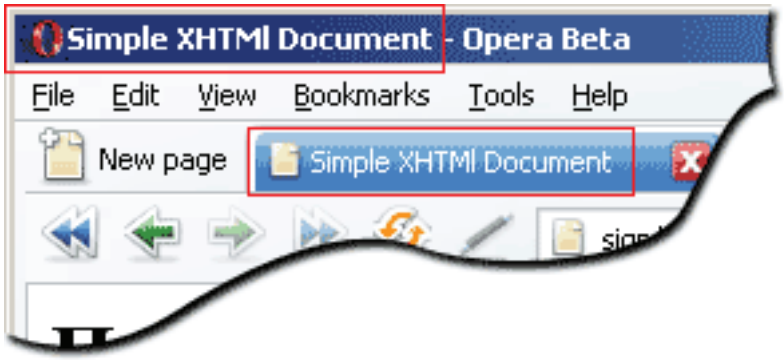

Рис. №1.4.2. Заголовок страницы. Opera, Firefox

Подбор краткого и информативного заголовка обеспечивает более быструю и простую ориентацию пользователя среди большого количества одновременно просматриваемых документов.

С остальными элементами раздела заголовка вы познакомитесь по мере освоения курса.

Тело документа - **<body>**

Тело документа представляет собой то, ради чего создается документ - его содержательную часть, то, что увидит конечный пользователь на странице. Здесь происходит разбиение текста страницы на структурные блоки (заголовки и абзацы), выполняется вставка изображений, таблиц, списков и прочих элементов содержания.

В приведенном выше примере документ содержит заголовок 1-го уровня (“h1”) и абзац (“p”).

5. Правила записи элементов и их атрибутов в XHTML.

Данный раздел познакомит вас с рядом правил, которым должны удовлетворять все элементы XHTML документа. По большому счету, это правила записи элементов XML. Однако,как уже неоднократно упоминалась ранее, XHTML документы также являются документами XML и подчинены тем же требованиям.

Введем ряд определений:

• **тэг** или **дескриптор** - специального вида текстовая метка в документе, отмечающая границы - начало или конец - некоторого элемента содержимого. Например, для обозначения начала абзаца, используется тэг , а для его завершения -

• **элементом** - называется совокупность открывающего и закрывающего тэга и его содержимого. Содержимым элемента может быть не только простой текст, а и вложенные в него другие элементы. Кроме того, существует ряд элементов принципиально не могущих иметь содержимого. Такие элементы подчиняются несколько отличным от традиционных правилам записи (см. ниже).

• **атрибут** - это свойство элемента. Записываются атрибуты в начальном тэге элемента и определяют внешний вид и логику использования элемента.

Теперь обо всем по порядку.

Элементы

Элементы - одна из наиболее важных сущностей XHTML документа. С помощью элементов производится разбиение страницы на структурные блоки и выполняется визуальное оформление документа.

Рассмотрим ряд правил. которым вы должны следовать при записи тэгов и самих элементов.

1. **Правило записи начального тэга.** Начальный тэг элемента должен быть записан в следующем виде:

<имя_элемента>

Сразу за символом "<" должно следовать имя элемента. Имена элементов должны быть обязательно записанны в нижнем регистре!

Примечание: В ранних версиях языка разметки HTML правила записи имен элементов несколько отличались. Например, не существовало принципиальной разницы между записью <BODY> и <body>. Браузеры до сих пор "понимают" устаревшую форму записи и не делают между ними различия. Однако, вы должны следовать требованиям современных стандартов, чтобы научиться создавать страницы не только привлекательно выглядящие в браузере, но и отвечающие современным представлениям о правильном оформлении XHTML документов.

В начальном тэге элемент может содержать один и более атрибутов (более подробно см. ниже):

<имя_элемента атрибут="значение" ...>

Имя элемента и имя первого атрибута должен разделять как минимум один пробельный символ.

2. Правило записи конечного тэга. Конечный тэг элемента должен быть записан в следующей форме:

...</имя_элемента>

Сразу за символом "<" должен следовать символ "/", а за ним - имя элемента. В закрывающем тэге больше ничего не должно содержаться. Перечисление здесь атрибутов элемента будет грубой ошибкой!

3. Правило записи элементов, не имеющих содержимого. В XHTML существуе ряд элементов, которые принципиально не могут иметь содержимого. К таким элементам относятся, например, элемент вставки рисунка - img или принудительный перенос - br. Безусловно, нет ошибки в такой форме записи:

<img ... ></img>

Однако, между открывающим и закрывающим тэгами элемента img никогда не может быть содержимого. Поэтому было принято соглашение об альтернативной, более компактной форме записи подобных элементов:

```
<img ... />
```

Последовательность " />" закрывает элемент так, как обыкновенный закрывающий тэг. Причем, между именем тэга или значением последнего атрибута (если таковые имеются) и символом "/" должен следовать хотя бы один пробел!

Примечание: Отсутствие пробела перед символом "/" не приведет к катастрофическим результатам при запуске страницы в браузере. Однако, при проверке документа HTML Validator выдает предупреждение об его отсутствии.

Атрибуты

Как было сказанно ранее, атрибуты - это способ указания свойств элемента. Существует, к примеру, возможность с помощью атрибутов повлиять на визуальное представление элемента, связав с ним стиль (об этомречь идет далее в уроке).

Атрибуты элемента перечисляются в его начальном тэге:

```
<имя_элемента атрибут="значение" атрибут="значение" ... >
```

При записи атрибута указывается пара имя="значение", причем значение атрибута обязательно заключается в одинарные (') или двойные (") кавычки. Между именем следующего атрибута и значением предыдущего обязательно должен следовать как минимум один пробел.

Ниже приводится перечень атрибутов, которые могут присутствовать в любом элементе:

id - уникальный идентификатор элемента

style - стиль элемента

class - класс стиля в таблице стилей CSS

title - всплывающая подсказка элемента

lang - код языка, применяемый к элементу и его содержимому

dir - задает направление размещения текста содержимого

Соблюдаем иерархию

XML является в первую очередь языком разметки структуры данных документа. Именно поэтому к иерархии, то есть к родительско-дочерним отношениям элементов в документе, предъявляются довольно жесткие требования. Элементы XHTML документа дожны быть полностью вложенными друг в друга. Ниже приводятся примеры правильного и неправильного вложения элементов:

```
Корректное вложение
<элемент1> ... <элемент2> ... </элемент2> ...</элемент1>
Некорректное вложение
<элемент1> ... <элемент2> ... </элемент1> ...</элемент2>
```

Соблюдать правильность вложения элементов вам поможет простое правило: первый открытый элемент в иерархии закрывается последним. То есть, для них действует правила стека - LIFO (Last In First Out - “последним вошел, первым вышел”).

Комментарии в тексте документа

Текст документа может содержать комментарии, текстовые фрагменты, не отображаемые браузером. Комментарии используются разработчиками для снабжения текста документа примечаниями, которые вносят ясность в структуру документа при его последующем разборе другими разработчиками.

Все комментарии XHTML должны записываться в следующем виде:

```
<!-- Это комментарий -->
<!--
   Комментарий может состоять
   из нескольких строк...
-->
```

6. Классификация элементов.

Набор элементов XHTML очень разнообразен. Для того, чтобы упорядочить наши будущие знания об элементах и их специфике, необходимо как нибудь их классифицировать. Именно этим мы и займемся в данном разделе урока.

Элементы блочной и линейной разметки

По способу форматирования содержимого элементы XHTML можно разделить на блочные и линейные. Грубо говоря, блочные элементы используются для выделения больших структурных блоков в тексте, таких как абзацы или заголовки. Линейные же элементв служат для выделения фрагментов текста внутри этих блоков. Для примера: абзац (“p”) - блочный элемент, выделение текста жирным (“b”) или курсивным (“i”) начертанием - линейные элементы.

Теперь более подробно. Любой блочный элемент по умолчанию размещается на странице таким образом, что занимает всю доступную ему ширину, а высота элемента уже зависит от содержимого. Несколько записанных подряд блочных элементов размещаются друг под другом, даже если им хватает места для размещения в одной строке:

Листинг №1.6.1: Простой пример блочной разметки

```
...
<body>
   <p>Первый абзац.</p>
   <p>И второй абзац.</p>
</body>
...
```

пример находится в папке: samples\ example_1.6.1.html

Обе фразы предыдущего примера будут выведены с новой строки.

Принципы разбиения текста на блоки

Ниже представлена графическая структура разметки сложного многоуровневого документа:

Рис. №1.6.1. Графическая структура разметки сложного многоуровневого документа

Любой сложный многоуровневый документ можно разбить на разделы. Раздел принято начинать заголовком. Это будет первый уровень разбиения. Каждый раздел, в свою очередь, можно поделить на более мелкие структурные единицы - подразделы (второй уровень). И так далее. Наименьшей структурной единицей блочной разметки текста является абзац.

Как же реализовать такую структурную разметку в терминах XHTML? Для этой цели используются следующие элементы:

<h1>, **<h2>**, ... , **<h6>** - элементы заголовков с первого по шестой уровень

<p> - абзац.

Абзац. Элемент "p"

Элемент **<p>** создает абзац в документе. Наличие закрывающего тэга является обязательным. Помимо стандартного набора атрибутов, элемент <p> может содержать атрибут align - выравнивание. Внимание: использование атрибута align считается устаревшим и заменяется стилем CSS, однако, для общего развития и сравнения мы, все же, его рассмотрим:

```
<p align="выравнивание"> ... </p>
```

Атрибут align может принимать такие значения:

left - по левой стороне (по умолчанию)

right - по правой стороне

center - по центру

justify - по ширине; последняя строка выравнивается по левой стороне

Заголовки. Элементы "h1" ... "h6"

Существует возможность выделять до шести уровней заголовка: с 1го по 6й. Заголовок 1го уровня - **<h1>**

является наиболее значимым и отображается большим по размеру шрифтом. Соответственно, заголовок бго уровня - **<h6>** - наименее значим (ниже только абзацы...) и отображается шрифтом наименьшим по размеру.

Как и элемент **<p>**, помимо общего набора атрибутов, заголовки могут использовать устаревший атрибут **align** для выравнивания текста заголовка относительно страницы. Поддерживаются те же значения выравнивания (left, right, center, justify).

Обработка пробельных символов. Элемент "pre"

Прежде чем перейти к изучению линейных элементов, поговорим о том, как процессор XHTML обрабатывает пробельные символы. Вы, вероятно, уже заметили, что слова, записанные с новой строки в редакторе, не обязательно отображаются с новой строки в браузере. Да и несколько идущих подряд пробелов выводятся как один. Все дело в том, что браузеры проводят предварительную обработку текста перед его форматированием и выводом.

Суть предварительной обработки состоит в следующем: несколько идущих подряд символов пробела или табуляции "сворачиваются" в один пробел; все символы "перевод строки и возврат каретки" (Enter) удаляются. Исключением является только ,преформатированный текст (см. ниже).

Таким образом, следующая фраза

```
<p>
Эта фраза
                будет выведена          одной
строкой с одним пробелом   между
каждым
словом!
</p>
```

... в браузере будет выглядеть вполне обычной:

Эта фраза будет выведена одной строкой с одним пробелом между каждым словом!

Однако, может возникнуть ситуация, когда все же будет необходимо написать каждую фразу некоторого текста с новой строки (например, расместить на странице некоторое стихотворение или программный код). В таких случаях исходное форматирование текста имеет довольно большое значение и его необходимо сохранить. Если попытаться сделать это с использованием элементов XHTML (например - создать столько абзацев, сколько строк в стихотворении или коде), можно сильно захламить текст страницы, так и не добившись требуемого результата.

Лучшим выходом может являться использование элемента **<pre>** - preformated, преформатированный. Этот элемент выводит свое содержимое на страницу с сохранением исходного форматирования текста.

Например:

```
<pre>
                Элемент pre сохраняет исходное
                форматирование фрагмента текста.

                                [Из справочника по XHTML]
</pre>
```

В результате получим:

<pre>
Элемент pre сохраняет исходное
форматирование фрагмента текста.

[Из справочника по XHTML]
</pre>

Линейные элементы.

Линейные элементы обладают, скажем так, меньшей сферой влияния по сравнению с блочными. Их можно использовать, например, для выделения фрагмента усиленного значения, некоторой фразы или слова в текстовом блоке. К примеру, важность слов “Первый” и “второй” в следующем примере, подчеркивается элементом **strong** и отображаться они будут полужирным начертанием:

Листинг №1.6.2: Простой пример линейной разметки

```
...
<body>
   <p><strong>Первый</strong> абзац.</p>
   <p>И <strong>второй</strong> абзац.</p>
</body>
...
```

пример находится в папке: samples\ example_1.6.2.html

Линейные элементы могут свободно размещаться в одной строке и вкладываться друг в друга.

По принципу разметки информации, линейные элементы можно разделить на следующие два типа:

1) элементы физического форматирования;

2)элементы логического форматирования.

Элементы логического форматирования привносят в HTML код семантику. То есть, по самому названию элемента можно судить о сути, смысле выделяемого фрагмента. При этом, физуальный эффект разметки играет второстепенную роль и даже может отсутствовать.

Примечание: Факт отсутствия визуального эффекта не должен вас беспокоить. В дальнейшем его можно добиться с помощью стилей CSS.

В противовес элементам логического форматирования, элементы физического форматирования на первое место выводят именно внешний вид содержимого. Такие элементы ровным счетом ничего не сообщают о типе выделяемой информации. Их использование считается нерекомендованным, хотя и приносит, на первый взгляд, куда большую пользу.

Элементы физического форматирования

<b> - элемент, выделяющий содержимое полужирным шрифтом. Специальные атрибуты отсутствуют.

Пример применения:

```
<p>Иногда что-то необходимо <b>выделить</b> из общего текста...</p>
```

<i> - элемент, выделяющий содержимое курсивом. Специальные атрибуты отсутствуют.

Пример применения:

```
<p>Иногда что-то необходимо <i>выделить</i> из общего текста...</p>
```

<u> - элемент, позволяющий выводить текст подчеркнутым. Специальные атрибуты отсутствуют. Данный элемент является устаревшим!

Пример применения:

```
<p>Иногда что-то необходимо <u>подчеркнуть</u>...</p>
```

<s> или **<strike>**- элементы, позволяющий выводить текст перечеркнутым. Специальные атрибуты отсутствуют. Данные элементы являются устаревшими!

Пример применения:

```
<p>Иногда что-то необходимо <s>зачеркнуть</s>...</p>
```

<big> - элемент, позволяющий выводить текст шрифтом больше базового. Специальные атрибуты отсутствуют.

Пример применения:

```
<p>Иногда что-то необходимо <big>выделить</big> из общего текста...</p>
```

<small> - элемент, позволяющий выводить текст шрифтом меньше базового. Специальные атрибуты отсутствуют.

Пример применения:

```
<p>Иногда что-то необходимо <small>выделить</small> из общего текста...</p>
```

<sub> - элемент, позволяющий выводить текст как нижний индекс (со смещением вниз от базовой линии). Специальные атрибуты отсутствуют.

Пример применения:

```
<p>Иногда что-то необходимо <sub>выделить</sub> из общего текста...</p>
```

<sup> - элемент, позволяющий выводить текст как верхний индекс (со смещением вверх от базовой линии). Специальные атрибуты отсутствуют.

Пример применения:

```
<p>Иногда что-то необходимо <sup>выделить</sup> из общего текста...</p>
```

<tt> - элемент, позволяющий выводить текст моноширным шрифтом (все символы имеют одинаковую ширину знакоместа). Специальные атрибуты отсутствуют.

Пример применения:

```
<p>Иногда что-то необходимо <tt>выделить</tt> из общего текста...</p>
```

пример находится в папке: samples\example_1.6.3.html.

Для достижения комбинированного эффекта при форматировании (например - создания полужирного подчеркнутого курсива) можно вкладывать элементы друг в друга:

```
<p>Иногда что-то необходимо <b><i><u>выделить</u></i></b> из общего текста...</p>
```

В примере выше слово “выделить” бутет выведено полужирным подчеркнуым курсивом.

Элементы логического форматирования

<abbr>, **<acronym>** - элементы, идентифицирующие выделенный текст как аббревиатуру. Специальные атрибуты отсутствуют. Рекомендуется добавлять атрибут title для расшифровки значения аббревиатуры. Визуальный эффект форматирования по умолчанию отсутствует.

Пример применения:

```
<p><acronym title=”World Wide Web Consortium”>W3C</acronym> - международная
организация, занимающаяся стандартизацией языков разметки.</p>
```

<address> - применяется для идентификации автора документа и указания адреса автора. Сюда же обычно помещаются и сведения об авторских правах, а также дата создания и последней модификации документа. Тег применяется обычно вначале или/и в конце документа. Специальные атрибуты отсутствуют. Выделенный текст по умолчанию отображается курсивом.

Пример применения:

```
<p><address>&copy; Артемов  АН, 2004г. http://alexart.od.ua</address></p>
```

<cite> - элемент, идентифицирующий выделенный текст как цитату или ссылку на другой содержащий

ее ресурс. Специальные атрибуты отсутствуют. Выделенный текст по умолчанию отображается курсивом.

Пример применения:

```
<p>Как сказал Марк Твен, <cite>”Бросить курить достаточно легко.
Лично я проделывал это неоднократно...”</cite></p>
```

<code> - элемент, идентифицирующий выделенный текст как фрагмент программного кода. Специальные атрибуты отсутствуют. Выделенный текст по умолчанию отображается моноширным шрифтом.

Пример применения:

```
<p><code>if (a*10 == b) printf (“%d”, a);</code></p>
```

<dfn> - элемент, используемый для выделения первого появления некоторого термина в документе. Специальные атрибуты отсутствуют. Выделенный текст по умолчанию отображается курсивом.

Пример применения:

```
<p><dfn>Абзац</dfn> - это блок текста, отмеченный элементом “p”.</p>
```

**** - emphasis (акцент). Используется для акцентирования внимания на некотором фрагменте текста. Специальные атрибуты отсутствуют. Выделенный текст по умолчанию отображается курсивом.

Пример применения:

```
<p><em>Абзац</em> - это блок текста, отмеченный элементом “p”.</p>
```

**** - используется для обозначения фрагмента текста усиленной важности. Специальные атрибуты отсутствуют. Выделенный текст по умолчанию отображается полужирным начертанием.

Пример применения:

```
<p><strong>Абзац</strong> - это блок текста, отмеченный элементом “p”.</p>
```

пример находится в папке: samples\example_1.6.4.html.

Резюме

Как можно заметить, некоторые элементы отличаются лишь названиями, форматируя свое содержимое одинаково (а то и вообще не форматируя). Используя стилевые правила CSS, вы сможете самостоятельно задать внешний вид любого элемента.

7. Стиль элемента – альтернатива устаревшим тегам и атрибутам.

Рассмотренный в предыдущем разделе набор элементов XHTML довольно широк, однако и он не позволяет решить любую задачу по оформлению web страниц. Многие элементы логического форматирования отображают заключенный в них текст одинаково, или без изменений и это одна из основных причин по которой вам необходимо использовать стиль CSS.

Однако, основным назначением стилей CSS является **отделение структуры документа от его представления** и расширение возможностей визуального оформления элементов XHTML.

Применение стиля

Детальному изучению Каскадных Таблиц Стилей посвящена отдельная часть нашего курса, в рамках же данного занятия нам предстоит лишь познакомиться с одним из способов применения стилей. Так же мы рассмотрим возможности правил CSS для шрифрового оформления текста страницы.

Наиболее простым способом применения стилей является использование атрибута **style**:

```
<элемент style=”правила стиля”> ...
```

Правила стиля записываются в виде пар “название:значение;” последовательно одно за другим. Например, если необходимо установить в заголовке 1го уровня размер текста равным 16 пунктам, а цвет сделать темно-красным, можно использовать следующий набор правил:

Листинг №1.7.1: Использование стиля

```
<body>
  <h1>Обыкновенный заголовок 1го уровня</h1>
  <h1 style=”font-size: 16pt; color: #660000;”>
    Заголовок 1го уровня, использующий стили.
  </h1>
</body>
```

пример находится в папке: samples\example_1.7.1.html.

Стили шрифтового оформления текста

CSS дают дизайнеру поистине великолепные возможности по форматированию шрифтов, в чем вы сами наглядно убедитесь, опробовав на практике следующие правила:

- **font-family**: название шрифта;
- **font-size**: размер;
- **font-weight**: normal | bold; - жирность начертания
- **font-style**: normal | italic; - курсив
- **color**: цвет символов.

Свойства **font-weight** и **font-style** используются довольнно тривиально и не стоят подробного рассмотрения. Они заменяют элементы b и i соответственно. Аспекты применения остальных свойств рассматриваются ниже.

Свойство font-family

Это свойство применяется для указания используемого шрифта или семейства шрифтов. Ниже показан пример назначения шрифта "Comic Sans MS" абзацу:

```
<p style="font-family: Comic Sans MS;"> ...
```

Текст абзаца будет выведен шрифтом "Comic Sans MS".

Совет: для того, чтобы узнать названия шрифтов, доступных вам для использования, раскройте панель управления Windows и выберите пункт "Шрифты" ("Fonts").

Однако, нет никаких гарантий, что используемый вами шрифт будет на компьютере остальных пользователей, ведь на каждом из них может быть установлен свой набор шрифтов, и есть лишь несколько

шрифтов, которые присутствуют на большинстве (но не на всех!) компьютерах.

Как же поступить в этом случае? Можно указать не один шрифт, а несколько, перечисляя их через “,”:

```
<p style=”font-family: Century Gothic, Tahoma, Arial;”> ...
```

Тогда браузер попытается отобразить текст первым шрифтом из списка. Если указанный шрифт на компьютере отсутствует, будет сделана попытка воспользоваться вторым из списка. Если и второй будет недоступен, браузер попробует использовать следующий — третий. Потом — четвёртый, пятый и так далее.

Но и в этом случае остается вероятность (пусть и небольшая) отсутствия всех перечисленных шрифтов на компьютере пользователя! Что же делать тогда? Необходимо завершать список указанием имени семейства, к которому относятся перечисленные шрифты. Ниже приводятся названия семейств и несколько относящихся к ним шрифтов:

• **serif** - антиква, шрифты с “засечками”. Наиболее распостраненный пример - Times New Roman. К этому семейству относятся так же Bookman Old Style, Garamond, Georgia, Minion Web.

• **sans-serif** - гельветика, шрифты “рубленные” или без “засечек”. Наиболее распостраненные шрифты этого семейства Arial, Tahoma, Verdana, Century Gothic, Franklin Gothic, Impact.

• **monospace** - моноширный, с одинаковой шириной знакоместа для каждого символа. Наиболее известные шрифты данного семейства - Courier, Courier New, Lucida Console, Monotype.com.

Итак, для того, чтобы избежать неприятной ситуации и нежелательной смены стилистики оформления шрифта ваших страниц на компьютерах других пользователей, список шрифтов завершаем указанием имени соответствующего семейства:

```
<p style=”font-family: Century Gothic, Tahoma, Arial, sans-serif;”> ...
```

Свойство **font-size**

С помощью данного свойства можно указать размер шрифта. Размер шрифта принято указывать в специальных единицах - пунктах(pt, 1pt = 1/72 дюйма). Однако, нет препятствий для использования других единиц измерения, таких как дюйм (in), сантиметр (cm), миллиметр (mm) и, наконец, пиксель (px). Название единицы измерения должно следовать сразу за величиной размера:

```
<p style=”font-size: 12pt;”> ...
```

Свойство **color**

Цвет символов текста можно указать, используя свойство color. При этом цвет вожет быть задан как названием, так и с помощью палитры web цветов:

```
<p style="color: red;"> ...
<p style="color: #FF0000;"> ...
```

В обоих случаях цвет текста будет красным.

Палитра web цветов с прилагается к уроку и вы можете ее найти в папке "WebColors" данного урока.

Еще пару слов об элементах

Уже сейчас, используя изученные стили оформления текста вы можете "заставить" выглядеть любой элемент логического (и не только) форматирования так, как вам захочется. Но иногда появляется необходимость, выделить фрагмент, к примеру, только цветом, какой элемент использовать? Кому назначать стиль? Специально для таких ситуаций, существует элемент ****. Это элемент - пустышка, обертка, которую можно сделать красивой, благодаря стилям: Листинг №1.7.2: Использование элемента span

```
<p style="color: #000066;">Предположим, что в тексте абзаца необходимо выделить
цветом <span style="color: #FF0000;">одно или несколько слов</span>.
Используйте для этих целей элемент
<span style="font-weight: bold; color: red;">"span"</span>.
</p>
```

пример находится в папке: samples/example_1.7.2.html

**** является линейным элементом, и не подходит для форматирования крупных текстовых блоков. Для этой цели необходимо использовать его блочный аналог - элемент ** **.

Домашнее задание

В качестве домашнего задания предлагается выполнить разметку текстового документа. Вам предстоит разбить текст на разделы, каждый из которых начинается заголовком, и выделить абзацы. Помимо этого, вы должны будете выделить в тексте логические сущности и отметить их подходящими элементами.

Текст для использования в домашнем задании взят из документов сайта компании W3C. **Результат должен выглядеть так:**

Дополнительная информация

Если Вы готовы учиться дальше, я приготовил дополнительный материал по углубленному курсу HTML и добавлению стилей.

Рекомендация W3C HTML 4.0 - это нормативная спецификация HTML. Однако, это техническая спецификация. Более просто информация изложена в книгах по HTML, например, "Raggett on HTML 4", опубликованная в 1998 г. издательством Addison Wesley. См. Также ***"Beginning XHTML"***, опубликована 2000 г. издательством Wrox Press, в которой вводится переформулировка HTML W3C в качестве приложения XML. XHTML 1.0 теперь является Рекомендацией W3C.

Удачи и пишите!

Дэйв Рэгетт <*dsr@w3.org*>