Содержание:

Введение

В современном мире, невозможно представить себе web-сайт безликим и простым. Каждый стремится придать своей странице как можно больше изюминок. Будь то анимация, красивый шрифт или изящный дизайн. За всем этим стоят языки разметки гипертекстов.

Технология World Wide Web (WWW), в переводе Всемирная паутина получила столь широкое распространение из-за простоты своих пользовательских интерфейсов. В технологиях WWW все ключевые понятия просматриваемого документа: слова, картинки - имеют возможность раскрыться новым документом, раскрывающим это понятие. Такой способ представления информации называется гипертекстом, а документы, представленные в таком виде – гипертекстовыми документами. Имея редактор гипертекстов, можно создать любую структуру рабочей среды, включая документацию, файлы, данные, картины, программное обеспечение и т.д., и это не будет новое программное обеспечение, а просто гипертекст. Для описания этих документов используются специальные языки - HTML (англ. вариант Hyper Text Markup Language), XML (Extensible Markup Language), SGML (Standard Generalized Markup Language).

Данные языки зародились в 90-х годах прошлого века. Однако, за прошедшее время успели активно развиться и получить широкое распространение. Интернет стал неотъемлемой частью нашей жизни, а web-разработка и дизайн престижной и востребованной профессией. А языки разметки гипертекста неотъемлемой частью работы web-разработчиков.

Именно поэтому актуальность данной курсовой работы столь высока.

Целью данной работы является изучить языки разметки гипертекста и понять их особенности и отличия.

Задачи:

- Рассмотреть основные сведения о каждом языке;
- Изучить структуру оформления web-страниц посредством каждого из языков;

- Понять в чем состоит принципиальное отличие каждого языка разметки гипертекста.

Структура: работа состоит из введения, трех глав, описывающих языки разметки гипертекста, заключения и списка использованной литературы.

1. Особенности языка разметки гипертекста HTML

1.1. О языке разметки HTML

HTML расшифровывается как HyperText Markup Language (язык разметки гипертекста):

• язык означает, что он может быть прочитан как человеком, так и компьютером;

• разметка означает, что написанный вами код помечается с помощью ключевых слов;

• гипертекст означает, что он использует HTTP как часть Интернета.

Как и любой язык, HTML поставляется с набором правил. Эти правила относительно простые и сводятся к определению границ, чтобы знать, где что-то начинается и где заканчивается.

В HTML реализована поддержка механизма специальных гипертекстовых ссылок, которые обеспечивают связь данного документа с другими документами. Последние могут находиться:

* на данном сайте, то есть в папке, содержащей все html-файлы, графику, звук, анимацию, видеофильмы, программы;

* вне сайта в других в папках на данном компьютере;

* в системе World Wide Web, то есть на других Web-серверах;

* в Internet на серверах других типов (FTP, Gopher).

Ранние версии HTML (HTML+ в 1994 году и HTML 2.0 в 1995) были разработаны на основе ранних работ Тима Бернерс-Ли с целью создания жизнеспособной системы

управления информацией. Однако когда Всемирная паутина завоевала мир, разработчики браузеров, в первую очередь таких, как Mosaic Netscape и Microsoft Internet Explorer, не стали ждать общих стандартов. Они дали людям то, что те хотели, создав множество элементов, улучшающих внешний вид страниц, но индивидуальных для каждого браузера. Это конкурентное противостояние получило название «Войны браузеров». В результате в конце 1990-х стало обычным делом создавать несколько разных версий сайта — по одной для каждого из популярных браузеров.

В 1996 году только что образованный консорциум Всемирной паутины (W3C) задал ориентир и выпустил первую Рекомендацию — HTML 3.2. Это собрание всех HTML-элементов, использовавшихся в то время. В него вошло множество презентационных расширений HTML, появившихся в результате соперничества браузеров, а также из-за отсутствия альтернативы в виде таблиц стилей. HTML 4.0 (1998) и HTML 4.01 (редакция с небольшими изменениями, которая заменила предшествующий стандарт в 1999) должны были вернуть HTML в нужное русло, подчеркнув разделение структуры и представления, и повысив доступность информации для пользователей с ограниченными возможностями.

Все задачи представления были переложены на новоиспеченные каскадные таблицы стилей (CSS), получавшие полную поддержку браузеров.

Примерно в то же время, когда разрабатывалась версия HTML 4.01, сотрудники консорциума Всемирной паутины осознали, что один язык разметки с ограниченными возможностями не получится использовать для описания всех видов информации (химических формул, математических уравнений, мультимедийных презентаций, финансовой информации и т. д.), которые можно распространять во Всемирной паутине. Их решение — XML (Расширяемый язык разметки), метаязык для создания языков разметки.

XML — это упрощенный вариант SGML (стандартного обобщенного языка разметки), главного метаязыка, который Тим Бернерс-Ли использовал для создания своего оригинального HTML-приложения. Но сам SGML оказался сложнее, чем требовалось для Всемирной паутины.

В консорциуме W3C предполагали, что Всемирная паутина будет развиваться на основе XML, и множество специализированных языков разметки будут использоваться совместно даже в пределах одного документа. Конечно, чтобы претворить это в жизнь, пришлось бы очень внимательно создавать разметку,

строго соблюдая синтаксис XML, чтобы исключить потенциальную путаницу.

Их первым шагом было переписать спецификацию HTML в соответствии с правилами XML, чтобы его можно было использовать вместе с другими XML языками. В результате появился XHTML (Расширяемый HTML). Первая версия, XHTML 1.0, почти идентичная спецификации HTML 4.01, содержит те же элементы и атрибуты, но имеет более жесткие требования относительно того, как должна выполняться разметка.

HTML 4.01 и его более строгий коллега XHTML 1.0 на основе XML, стали краеугольным камнем движения по развитию веб-стандартов.

Но консорциум Всемирной паутины не останавливается на достигнутом. Не забывая идею создания Всемирной паутины на основе XML, он начал работу над XHTML 2.0 — еще более смелой, чем HTML 4.01, попыткой заставить все работать «правильно». Проблема в том, что этот язык оказался несовместим со старыми стандартами и поведением браузеров. Процесс написания и утверждения затянулся на годы.

Без реализации в браузерах создание XHTML 2.0 застопорилось.

В 2014 году был запущен HTML 5.

Самая последняя разработка рабочей группы по HTML в W3C - рабочий проект HTML5.2 (2017 год). Основной ее целью является улучшить язык, поддерживающий работу с новейшими мультимедийными приложениями, при этом сохраняется легкость чтения кода для человека и ясность исполнения для компьютеров и приспособлений (web browsers, parsers и т. д.). HTML5 включает в себя не только HTML 4, но и XHTML 1, а также DOM2HTML(особенно JavaScript). HTML 5 — также попытка определить единый язык разметки, который мог бы быть написан как и в HTML, так и в XHTML и был бы синтаксически корректен. Включает в себя детальные модели обработки, чтобы поддерживать больше взаимодействующих процессов; расширяет, улучшает и рационализирует разметку, пригодную для документов, и вводит разметку и API для сложных веб-приложений.

HTML 5 представляет несколько новых элементов и атрибутов, которые часто используются на современных веб-сайтах. Некоторые из них семантически заменены для общего использования базовых блоков и строковых элементов , например, <nav> (блок навигации по сайту), <footer>(обычно обращение к нижней части страницы или последней строке HTML кода), или <audio> и <video>

вместо <object>. Некоторые элементы, которые можно было использовать в HTML 4.01, были исключены, например, представляемые элементы, такие как <font> и , чьи эффекты выполняются с помощью Cascading Style Sheets (Каскадная таблица стилей). Также в поведении веб снова заострено внимание на важности скриптов DOM(например, Javascript).

Синтаксис HTML 5 больше не базируется на SGML несмотря на подобие его разметки. Однако он был разработан как обратный аналогу, с общим анализом более старых версий HTML. Идет новая вводная строка, которая выглядит так же, как и в SGML в описании типа документа, <!DOCTYPE html> , которая запускает соответствующий стандартам предоставленный метод. С 5 января 2009 года HTML 5 также включает в себя Web Forms 2.0, ранее выделенный спецификацией WHATWG.

В дополнении к определению разметки HTML 5 устанавливает скриптовый Интерфейс прикладного программирования(API). Существующий интерфейс DOM расширен и фактически особенности зарегистрированы.

1.2. Основные составляющие HTML-документа

Документ состоит из текстов, графики, таблиц и других объектов, которые представляют собой содержимое документа. Программа просмотра использует при этом теги, которые записаны в HTML-документе для задания структуры расположения объектов и их внешнего вида. Чаще всего HTML-теги записываются парами (начальный и конечный теги), между которыми размещаются текст и другие объекты документа. Имя конечного тега идентично имени начального, но перед именем конечного тега ставится косая черта (/), так называемый слэш. Оформление HTML-документа просто: он начинается тегом и заканчивается тегом . Имя тега может быть записано как строчными, так и прописными буквами.

То, что можно увидеть в угловых скобках < и > называется тегами HTML. Они определяют, где что-то начинается и где заканчивается.

Каждый из тегов несёт определённый смысл. Для примера: <p> обозначает абзац текста.

Как правило, они идут парами:

открывающий тег <p> определяет начало абзаца;

закрывающий тег определяет его конец.

Единственным различием между открывающим и закрывающим тегами является слэш /, который предшествует имени тега.

При объединении открывающего, закрывающего тегов и всего между ними, получаем элемент HTML. Строка целиком представляет собой элемент HTML, который использует теги HTML и .

Если просмотреть пример кода в браузере, то можно заметить, что теги HTML в браузере не отображаются. Они читаются только браузером, чтобы знать, какой тип контента был написан.

Целью тегов HTML является передача смысла документу. В зависимости от написанного содержимого, можно выбрать подходящий элемент, соответствующий смыслу текста.

Виды тегов:

одиночные (непарные) теги <tag>. Например тег
, который используется для перехода на новую строку

теги-контейнеры <tag></tag>. Например тег . После открывающего тега весь текст будет полужирным, пока тег не будет закрыт.

Тег может иметь атрибуты. Атрибуты уточняют действие тега. Атрибут всегда ставится в открывающем теге, т.к. браузер считывает информацию слева направо сверху вниз. Для некоторых тегов применение атрибутов обязательно. Так например тег вставки изображения <img> должен иметь атрибут, в котором будет указан путь к файлу изображения, иначе изображение не будет отображаться на странице <img src="путь_к_изображению">. Некоторые теги могут использоваться вообще без атрибутов.

Теги можно писать как в верхнем <TAG>, так и в нижнем <tag> регистре, для браузера это значения не имеет. Также не имеет значения, писать ли весь текст в одну строку или начинать каждый абзац или предложение с новой строки. Для читабельности кода рекомендуется всегда использовать один регистр (например нижний), начинать каждый новый элемент с новой строки и вставлять в код комментарии.

Пробелы:

пробел ставится обязательно между тегом и атрибутом

пробел недопустим между атрибутом и его значением

в тексте html-документа несколько пробелов воспринимаются как один, для того, чтобы поставить несколько пробелов подряд, надо использовать специальные символы

Диапазон элементов достаточно широк, чтобы он подходил и для материалов общего назначения (например, абзацы или списки) и для более конкретного содержимого, вроде <output> (для отображения результата вычисления) или <progress> (для отображения хода выполнения задачи).

Структурные элементы позволяют организовать основные части страницы. Они обычно содержат другие элементы HTML.

Типичная веб-страница может включать:

<header> в качестве первого элемента страницы, который может включать в себя логотип и слоган;

<nav> в качестве списка ссылок, которые ведут на разные страницы сайта;

<h1> в качестве заголовка страницы;

<article> в качестве основного содержимого страницы, вроде статьи блога;

<footer> в качестве последнего элемента страницы, расположенного внизу.

Внутри структурных элементов обычно можно наблюдать текстовые элементы, призванные определить цель содержимого.

В основном используется:

<p> для абзацев;

 для (неупорядоченных) списков;

 для (упорядоченных) списков;

 для отдельных пунктов списка;

<blockquote> для цитат.

Поскольку текстовые элементы могут быть длинными, но с разным содержанием, строчные элементы позволяют различать части текста.

Есть много строчных элементов, но чаще всего можно столкнуться со следующими:

 для важных слов;

 для выделенных слов;

<a> для ссылок;

<small> для менее важных слов;

<abbr> для аббревиатур, вроде W3C.

Когда ни один семантический элемент не подходит для содержимого, но всё ещё необходимо вставить элемент HTML (в целях группирования или стилизации), то можно остановиться на одном из двух общих элементов:

 для блочных элементов;

 для строчных элементов.

Хотя эти элементы HTML на самом деле не несут какого-либо смысла, они пригодятся при использовании CSS.

Существует около 100 семантических элементов HTML на выбор.

Написание кода HTML происходит в редакторе вроде Notepad++. Написанный код необходимо сохранить с разрешением .html.

Для открытия HTML-документов используется браузер, вроде Firefox.

HTML-документ (или веб-страница, что означает то же самое) требует определённой структуры для того, чтобы стать валидным.

Почему важно заботиться о валидации HTML-документа?

Правильность: валидный документ корректно отображается в браузере.

Отладка: некорректный код HTML может вызвать ошибки, сложные для выявления.

Поддержка: валидный документ легче обновлять позже, даже кому-то другому.

Первой информацией которая указывается - тип HTML-документа — доктайп.

Доктайп можно сравнить с версией автомобиля на протяжении многих лет: Ford Fiesta, купленный в 1986 году, был Fiesta 2. Если вы покупаете его сегодня, то это Fiesta 7.

Раньше сосуществовало несколько версий HTML (XHTML и HTML 4.01 были конкурирующими стандартами). В настоящее время нормой является HTML5.

Чтобы сообщить браузеру, что HTML-документ представляет собой HTML5, документ начинается со следующей строки:

<!DOCTYPE html>

HTML5 не упоминает цифру 5 потому что в W3C решили, что предыдущие определения доктайпа были слишком запутанными и воспользовались возможностью, чтобы упростить его, удалив упоминание о версии HTML.

Элемент <html>

Помимо строки с доктайпом, весь HTML-документ должен располагаться внутри элемента <html>:

<!DOCTYPE html>

<html>

<!-- Остальная часть кода HTML -->

Из всех видов это самый короткий доктайп, его легко запомнить и набирать по памяти. Изменения претерпели и другие теги, так, у тега <html> нет атрибу-та xmlns, а кодировка документа сократилась до такой записи.

<meta charset="utf-8">

Впрочем, старый способ указать кодировку также остался.

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

Атрибут type у тега <script> и <style> можно опустить, браузер автоматически понимает содержимое этих тегов и ему уже не требуется явно об этом напоминать. Простейший код приведён в примере 1.

Пример 1. Код на HTML5

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Пример страницы</title>
<style>
p { color: navy; }
</style>
</head>
<body>
<p>Страница на HTML5</p>
</body>
</html>
</html>
```

<html> технически является предком всех элементов HTML.

<head>

Как атрибуты несут дополнительную информацию для элемента HTML, так и элемент <head> несёт дополнительную информацию для всей веб-страницы.

Например, заголовок страницы (отображается на вкладке) находится в <head>:

```
<head>
<title>Курсовая работа</title>
</head>
```

Следующие элементы HTML могут появляться в <head> и только в <head>:

<link>

<meta>

<style>

<body>

В то время как <head> содержит только метаданные, не предназначенные для отображения вообще (за исключением <title>), то элемент <body> это место, где пишется всё содержимое. Всё внутри <body> будет отображаться в окне браузера.

1.3. Валидность HTML-документа

Объединив все эти требования, можно написать простой и валидный HTML-документ:

JSFiddle

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>MarkSheet</title>

<meta name="description" content="Простое руководство по HTML">

</head>

<body>

Hello, world!

</body>

</html>

Если просмотреть этот пример в браузере, то можно увидеть, что:

«MarkSheet» написано на вкладке браузера;

«Hello, world!» — это единственный текст, отображаемый в окне, потому что это единственное содержимое <body>.

Можно дополнять код комментариями без нарушения отображения страницы браузером: комментарий начинается с <!-- и заканчивается -->.

Сборка страницы происходит мгновенно, поэтому она появляется так, как будто вся страница загружается сразу. При медленных соединениях, или на более медленных компьютерах, или если страница содержит большое количество графики, процесс сборки более заметен, поскольку изображения появляются позже текста. Иногда странице даже требуется перезагрузка по мере появления новых изображений (хотя вы можете сконструировать свои страницы так, чтобы этому воспрепятствовать).

Спецификации HTML, предшествовавшие HTML5, содержали документацию только по элементам, атрибутам и значениям, допустимым в языке. Это хорошо для простых текстовых документов, но разработчики HTML5 планировали облегчить процесс создания веб-приложений, для которых требуются сценарии и программирование. По этой причине, чтобы облегчить взаимодействие с приложением, HTML5 также определяет ряд новых API-интерфейсов.

Интерфейс прикладного программирования (Application Programming Interface, API) — это задокументированный набор команд, имен данных и так далее, который позволяет одному программному приложению общаться с другим. Например, разработчики Twitter указали имена каждого типа данных (пользователи, твиты, метки и т. д.) и методы для доступа к ним в документе API-интерфейса (dev.tw itter.com /docs), что позволяет другим разработчикам добавлять каналы и элементы Twitter в свои продукты.

Поэтому существует так много программ и виджетов, поддерживающих сеть Twitter.

Интернет-магазин Amazon.com также демонстрирует информацию о своих товарах через API-интерфейс. В самом деле, все издатели признают, что очень хорошо, если контент доступен таким образом.

Можно сказать, что API-интерфейсы сейчас очень популярны.

Идея состоит в том, что если браузеры изначально предлагают эти функции со стандартизованными наборами методов для доступа к ним, разработчики могут создавать замечательные вещи и рассчитывать на то, что они будут работать во всех браузерах, так же, как мы сегодня привыкли к возможности вставлять изображения на страницы. Конечно, пройдет еще немало времени, пока поддержка этих передовых функций станет повсеместной, но прогресс налицо. Некоторые API-интерфейсы содержат компонент разметки, такой как вложение мультимедийного контента с новыми HTMЬ5-элементами video и audio. Другие выполняются полностью «за кадром» с применением сценариев JavaScript или серверных компонентов, например, создание веб-приложений, которые работают даже при отсутствии подключения к Интернету (A PIинтерфейс механизма кэширования данных веб-приложений).

Консорциум Всемирной паутины и сообщество WHATWG трудятся над множеством API-интерфейсов, которые можно будет использовать с веб-приложениями, и все они находятся в той или иной стадии завершения и реализации.

Большинство из них имеют свои собственные спецификации, отдельно от HTML5, но они, как правило, включены в общую спецификацию HTML5, которая охватывает все веб-приложения.

2. Особенности языка разметки гипертекста XML

2.1. Общее описание XML

XML - Extensible Markup Language, то есть Расширяемый Язык Разметки, возник в результате развития языка HTML (HyperText Markup Language, языка разметки гипертекста). Тем не менее было бы грубой ошибкой воспринимать его лишь как некую усовершенствованную версию языка HTML. По сути, язык XML представляет собой новое поколение языков разметки. Здесь следует иметь в виду 3 момента, принципиально отличающих XML от HTML и его предшественников (GML - Generalized Markup Language, SGML - Standard General Markup Language):

XML, в отличие от HTML, не имеет предопределенных тэгов - точнее, каждый разработчик может создавать свои собственные XML-тэги - столько, сколько нужно. Количество таких тэгов практически неограничено. Таким образом, XML является метаязыком, позволяющим создавать другие языки разметки, такие как, например,

HTML.

По мере развития языка HTML количество тэгов быстро увеличивалось. В конце концов их число достигло "критического значения" - разработчикам web-документов стало трудно запоминать все новые и новые тэги, но еще в худшем положении оказались разработчики браузеров - им приходится создавать все новые версии браузеров, которые "понимали бы" новые тэги. Более "умные" браузеры становятся и большими по объему, предъявляют все возрастающие требования к компьютерам, на которых они используются. Дело усугубляется тем, что в последнее время все большую популярность приобретают карманные устройства (в частности, они все шире используются в электронной коммерции), с ограниченным объемом памяти и "слабенькими" экранами, а потому, браузеры, используемые на них, имеют лишь очень ограниченные возможности. Язык XML, не имеющий определенной заранее системы тэгов, позволяет решить эту проблему. "Платой" за универсальность является большая строгость оформления web-документов. Правила оформления XML-документов просты:

недопустимы незакрытые контейнеры тэгов (но можно объединять открывающий и закрывающий тэги в одном, например:
)

"вложенные" контейнеры не могут "перекрываться"

строчные и прописные буквы воспринимаются как разные символы

в качестве названий тэгов нельзя использовать ключевые слова

в названиях тэгов нельзя использовать пробелы, знаки пунктуации, круглые, квадратные и фигурные скобки

знак подчеркивания (_) и цифры могут встречаться в названиях тэгов, но цифра не может быть перым символом названия тэга

(При необходимости использовать несколько слов в качестве названия тэга их следует писать слитно, начиная каждое слово с большой буквы.)

Документы, отвечающие этим правилам, называются well-formed documents.

XML служит для описания структуры данных, главным образом, иерархических структур.

Одной из основных тенденций развития web-технологий является разделение данных, структуры документа и его стилевого оформления. Как известно, одним из способов обособления данных от структуры документа является динамическое связывание СУБД с web-документами через интерфейс ODBC (Open DataBase Connectivity). Обособление стилей достигается за счет использования каскадных таблиц стилей. XML позволяет описывать нереляционные базы данных. Поскольку тэги могут создаваться разработчиком, их названия обычно характеризуют смысл данных.

XML, как средство описания структуры данных, обеспечивает обмен данными между различными приложениями, выступая, таким образом, в качестве своеобразного "клея".

Значение этой "связующей" функции XML трудно переоценить. Благодаря возможности обмена данными между различными приложениями web-технологии "выходят" на качественно новый уровень.

За неполный год своего официального существования язык XML привлек к себе уже достаточно много внимания со стороны разработчиков и пользователей Интернет. Сегодня количество приверженцев этой новой технологии возрастает также стремительно, как и число сообщений об очередных взятых ею преградах на пути к всеобщему признанию. Несмотря на то, что XML очень молод (международная организация W3C утвердила спецификацию "Extensible Markup Language(XML) 1.0" в начале февраля 1998 г) и отдельные компоненты этого языка находятся еще в стадии доработки, уже сегодня появляются новые языки, созданные на основе XML, возникают многочисленные Web-сервера, использующие эту технологию для организации хранящейся на них информации.

XML (Extensible Markup Language) - это язык разметки, описывающий целый класс объектов данных, называемых XML- документами. Этот язык используется в качестве средства для описания грамматики других языков и контроля за правильностью составления документов. Т.е. сам по себе XML не содержит никаких тэгов, предназначенных для разметки, он просто определяет порядок их создания. Таким образом, если, например, мы считаем, что для обозначения элемента rose в документе необходимо использовать тэг <flower>;, то XML позволяет свободно использовать определяемый нами тэг и мы можем включать в документ фрагменты, подобные следующему:

<flower>rose</flower>

Набор тэгов может быть легко расширен. Если, предположим, мы хотим также указать, что описание цветка должно по смыслу идти внутри описания оранжереи, в которой он цветет, то просто задаем новые тэги и выбираем порядок их следования:

<conservatory>

<flower>rose</flower>

</conservatory>

Если мы хотим посадить туда еще несколько цветочков, то должны внести следующие изменения:

<conservatory>

<flower>rose</flower>

<flower>tulip</flower>

<flower>cactus</flower>

</conservatory>

Как видно, сам процесс создания XML документа очень прост и требует от нас лишь базовых знаний HTML и понимания тех задач, которые мы хотим выполнить, используя XML в качестве языка разметки. Таким образом, у разработчиков появляется уникальная возможность определять собственные команды, позволяющие им наиболее эффективно определять данные, содержащиеся в документе. Автор документа создает его структуру, строит необходимые связи между элементами, используя те команды, которые удовлетворяют его требованиям и добивается такого типа разметки, которое необходимо ему для выполнения операций просмотра, поиска, анализа документа.

Еще одним из очевидных достоинств XML является возможность использования его в качестве универсального языка запросов к хранилищам информации. Сегодня в глубинах W3C находится на рассмотрении рабочий вариант стандарта XML-QL(или XQL), который, возможно, в будущем составит серьезную конкуренцию SQL. Кроме того, XML-документы могут выступать в качестве уникального способа хранения данных, который включает в себя одновременно средства для разбора информации и представления ее на стороне клиента. В этой области одним из перспективных

направлений является интеграция Java и XML - технологий, позволяющая использовать мощь обеих технологий при построении машинно-независимых приложений, использующих, кроме того, универсальный формат данных при обмене информации.

XML позволяет также осуществлять контроль за корректностью данных, хранящихся в документах, производить проверки иерархических соотношений внутри документа и устанавливать единый стандарт на структуру документов, содержимым которых могут быть самые различные данные. Это означает, что его можно использовать при построении сложных информационных систем, в которых очень важным является вопрос обмена информацией между различными приложениями, работающими в одной системе. Создавая структуру механизма обмена информации в самом начале работы над проектом, менеджер может избавить себя в будущем от многих проблем, связанных с несовместимостью используемых различными компонентами системы форматов данных.

Также одним из достоинств XML является то, что программы-обработчики XML-документов не сложны и уже сегодня появились и свободно распространяются всевозможные программные продукты, предназначенные для работы с XML-документами. На основе XML уже сегодня созданы такие известные специализированные языки разметки, как SMIL, CDF, MathML, XSL, и список рабочих проектов новых языков, находящихся на рассмотрении W3C, постоянно пополняется.

2.2. Структура XML-документа

Структура документа

Простейший XML- документ может выглядеть так, как это показано в Примере 1

<?xml version="1.0"?>

<list_of_items>

<item id="1"><first/>Первый</item>

<item id="2">Второй <sub_item>подпункт 1</sub_item></item>

<item id="3">Третий</item>

<item id="4"><last/>Последний</item>

</list_of_items>

Стоит обратить внимание на то, что этот документ очень похож на обычную HTML-страницу. Также, как и в HTML, инструкции, заключенные в угловые скобки называются тэгами и служат для разметки основного текста документа. В XML существуют открывающие, закрывающие и пустые тэги (в HTML понятие пустого тэга тоже существует, но специального его обозначения не требуется).

Тело документа XML состоит из элементов разметки (markup) и непосредственно содержимого документа - данных (content). XML - тэги предназначены для определения элементов документа, их атрибутов и других конструкций языка.

Любой XML- документ должен всегда начинаться с инструкции <?xml?>, внутри которой также можно задавать номер версии языка, номер кодовой страницы и другие параметры, необходимые программе-анализатору в процессе разбора документа

Правила создания XML- документа

В общем случае XML- документы должны удовлетворять следующим требованиям:

В заголовке документа помещается объявление XML, в котором указывается язык разметки документа, номер его версии и дополнительная информация

Каждый открывающий тэг, определяющий некоторую область данных в документе обязательно должен иметь своего закрывающего "напарника", т.е., в отличие от HTML, нельзя опускать закрывающие тэги

В XML учитывается регистр символов

Все значения атрибутов, используемых в определении тэгов, должны быть заключены в кавычки

Вложенность тэгов в XML строго контролируется, поэтому необходимо следить за порядком следования открывающих и закрывающих тэгов

Вся информация, располагающаяся между начальным и конечными тэгами, рассматривается в XML как данные и поэтому учитываются все символы форматирования (т.е. пробелы, переводы строк, табуляции не игнорируются, как в HTML)

Если XML- документ не нарушает приведенные правила, то он называется формально-правильным и все анализаторы, предназначенные для разбора XML-документов, смогут работать с ним корректно.

Однако кроме проверки на формальное соответствие грамматике языка, в документе могут присутствовать средства контроля над содержанием документа, за соблюдением правил, определяющих необходимые соотношений между элементами и формирующих структуру документа. Например, следующий текст, являясь вполне правильным XML- документом, будет абсолютно бессмысленным:

<country><title>Kazakhstan</title><city><title>Almaty</country></title></city>

Для того, чтобы обеспечить проверку корректности XML- документов, необходимо использовать анализаторы, производящие такую проверку и называемые верифицирующими.

На сегодняшний день существует два способа контроля правильности XML-документа: DTD - определения(Document Type Definition) и схемы данных(Semantic Schema).

Конструкции языка

Содержимое XML- документа представляет собой набор элементов, секций CDATA, директив анализатора, комментариев, спецсимволов, текстовых данных. Рассмотрим каждый из них подробней.

Элементы данных

Элемент - это структурная единица XML- документа. Заключая слово rose в в тэги <flower> </flower> , мы определяем непустой элемент, называемый <flower>, содержимым которого является rose. В общем случае в качестве содержимого элементов могут выступать как просто какой-то текст, так и другие, вложенные, элементы документа, секции CDATA, инструкции по обработке, комментарии, - т.е. практически любые части XML- документа.

Любой непустой элемент должен состоять из начального, конечного тэгов и данных, между ними заключенных. Например, следующие фрагменты будут являться элементами:

<flower>rose</flower>

<city>Almaty</city>

а эти - нет:

<rose>

<flower>

rose

Набором всех элементов, содержащихся в документе, задается его структура и определяются все иерархическое соотношения. Плоская модель данных превращается с использованием элементов в сложную иерархическую систему со множеством возможных связей между элементами.

В XML документе, как правило, определяется хотя бы один элемент, называемый корневым и с него программы-анализаторы начинают просмотр документа.

В некоторых случаях тэги могут изменять и уточнять семантику тех или иных фрагментов документа, по разному определяя одну и ту же информацию и тем самым предоставляя приложению-анализатору этого документа сведения о контексте использования описываемых данных. Например, прочитав фрагмент <city>Holliwood</city> мы можем догадаться, что речь в этой части документа идет о городе, а вот во фрагменте <restaurant>Holliwood</restaurant> - о забегаловке.

Комментарии

Комментариями является любая область данных, заключенная между последовательностями символов <!-- и --> Комментарии пропускаются анализатором и поэтому при разборе структуры документа в качестве значащей информации не рассматриваются.

Атрибуты

Если при определении элементов необходимо задать какие-либо параметры, уточняющие его характеристики, то имеется возможность использовать атрибуты эдлемента. Атрибут - это пара "название" = "значение", которую надо задавать при определении элемента в начальном тэге. Пример:

<color RGB="true">#ff08ff</color>

<color RGB="false">white</color>

или

<author id=0>Ivan Petrov</author>

Примером использования атрибутов в HTML является описание элемента :

Black

Специальные символы

Для того, чтобы включить в документ символ, используемый для определения каких-либо конструкций языка (например, символ угловой скобки) и не вызвать при этом ошибок в процессе разбора такого документа, нужно использовать его специальный символьный либо числовой идентификатор. Например, < , > " или $(десятичная форма записи), &#x1a (шестнадцатеричная) и т.д. Строковые обозначения спецсиволов могут определяться в XML документе при помощи компонентов (entity).

Директивы анализатора

Инструкции, предназначенные для анализаторов языка, описываются в XML документе при помощи специальных тэгов - <? и ?>;. Программа клиента использует эти инструкции для управления процессом разбора документа. Наиболее часто инструкции используются при определении типа документа (например, <? Xml version=”1.0”?>) или создании пространства имен.

CDATA

Чтобы задать область документа, которую при разборе анализатор будет рассматривать как простой текст, игнорируя любые инструкции и специальные символы, но, в отличии от комментариев, иметь возможность использовать их в приложении, необходимо использовать тэги <![CDATA] и]]>. Внутри этого блока можно помещать любую информацию, которая может понадобится программе-клиенту для выполнения каких-либо действий (в область CDATA, можно помещать, например, инструкции JavaScript). Естественно, надо следить за тем, чтобы в области, ограниченной этими тэгами не было последовательности символов]].

3. Особенности языка разметки гипертекста SGML

3.1. Общая характеристика SGML

SGML (Standard Generalized Markup Language) принят в 1986 году в качестве международного стандарта для определения независимых от устройств ввода/вывода, независимых от вычислительной среды методов представления текстов в электронной форме. Более точно, SGML - это метаязык, то есть средство формального описания языка, в данном случае, языка разметки.

Три характеристики SGML отличают его от прочих языков разметки:

- Описательная разметка

Система с описательной разметкой использует коды разметки, которые просто предоставляют названия для категоризации частей документа. Коды разметки, такие как <para> или \end{list}, просто идентифицируют порцию документа и утверждают, что "она является параграфом", или что "это - конец последним начатого списка" и т.п. С другой стороны, система с процедурной разметкой определяет, какая обработка должна выполняться в конкретной точке документа: "в этом месте вызвать процедуру PARA с параметрами 1, b и x", или "передвинуть левую границу на 2мм левее, правую границу -- на 2мм правее, пропустить одну строку и встать на новую левую границу" и т.п. В SGML инструкции, необходимые для обработки документа с какой-либо конкретной целью (например, для форматирования), четко отделяются от описательной разметки, которая встречается внутри документа. Обычно они собраны вне документа в отдельных процедурах или программах.

С описательной, а не процедурной, разметкой один и тот же документ может быть обработан разнообразными программами, каждая из которых может применять различные инструкции обработки к тем его частям, которые она считает важными. Например, программа анализа содержимого может полностью игнорировать сноски, тогда как программа форматирования может извлекать и собирать их для печати в конце каждой части. Различные виды инструкций обработки могут ассоциироваться с одной и той же частью файла. Например, одна программа может извлекать из документа фамилии людей и географические названия для создания индекса или базы данных, тогда как другая, обрабатывающая тот же самый текст, может печатать фамилии и названия отличающимся шрифтом.

- Типы документов

SGML вводит понятие типа документа, и, соответственно, определения типа документа (document type definition, DTD). Документы считаются типизированными, так же, как и другие обрабатываемые компьютерами объекты. Тип документа формально определяется его составными частями и их структурой. Определение, например, отчета может быть таким, что он состоит из заголовка и, возможно, автора, за которыми следует аннотация и последовательность одного или более абзацев. Любой документ в отсутствие заголовка, в соответствии с этим формальным определением, не будет формально являться отчетом, так же как не будет им являться и последовательность абзацев, за которой следует аннотация, невзирая на то, насколько похож на отчет такой документ с точки зрения читателя-человека.

Поскольку документы относятся к известным типам, можно использовать специальную программу, называемую анализатором (parser), для того, чтобы обработать документ, утверждающий, что он относится к конкретному типу, и проверить, действительно ли все элементы, требуемые для данного типа документов, присутствуют и находятся в правильной последовательности. Что еще более важно, разные документы одного типа могут обрабатываться унифицированным образом. Можно писать более интеллектуальные программы, использующие знания, заключенные в информационной структуре документа.

- Независимость данных

Основная цель проектирования SGML была в создании гарантий того, что документ, закодированный согласно его положениям, будет переносимым с одной аппаратной и программной среды в другую без потери информации. Два его свойства, описанных выше, отвечают этому требованию на абстрактном уровне; третье свойство - на уровне строчек байтов (символов), которые составляют документ. SGML предоставляет обобщенный механизм строковой подстановки, то есть, простой машинно-независимый способ указания, что конкретная строка символов в документе в момент обработки документа должна заменяться на некоторую другую строку. Одно очевидное применение этому механизму - обеспечение единой терминологии; другое, и более значительное, - противодействие известной неспособности разных компьютерных систем понимать наборы символов друг друга, например, одной системе представлять все графические символы, необходимые приложению, путем описательного отображения непередаваемых символов. Строки, определяемые этим механизмом, называются сущностями (entities).

Преимущества:

Что дает использование SGML в реальной жизни? Возможность производить документы в любом требуемом виде. Все использованные программы - бесплатны и распространяются в исходных текстах, что позволит установить их на всех используемых системах.

Продуктивность:

Четко разделенные процессы ввода информации и ее форматирования позволяют автору сосредоточиться на изложении мыслей, не отвлекаясь на двигание текста по экрану и подбор стилей.

Единая стилистика:

Легко выдерживать различные документы в едином стиле, используя единую терминологию. Если стиль или термины нужно изменить, это делается разом во всех документах, не трогая их содержимое.

Повторное использование:

Часть документа, оформленная в виде SGML-элемента, может переноситься в другие документы, легко повторяться в разных местах текста.

Долговечность информации:

Из-за того, что SGML - простой и стандартный формат хранения данных, отсутствует необходимость переформатировать их ввиду устаревания аппаратной или программной платформы. Информация просто доступна навсегда. Она несет с собой все необходимое для создания документа.

Управление данными:

С SGML можно определять информационные элементы и манипуляции с ними с произвольной степенью детальности. Размеченные элементы могут иметь атрибуты, определяющие характеристики и свойства элементов. Эта информация не предназначена для печати, но может помочь в управлении элементами данных. Например, атрибут ID (идентификатор) может уникальным образом идентифицировать один абзац, или целый раздел, примечание, иллюстрацию, задание, - любой элемент, как в этом примере:

<para id=431>Информация</para>

Так как идентификаторы являются машинно-читаемыми, они могут связывать между собой информацию и использоваться для разнообразного управления ей. Например:

- Контролировать безопасность доступа к информации, позволяя только определенным людям просматривать или изменять ее.
- Автоматизировать перемещение информации - например, обновление данных в одном месте может инициировать обновление той же информации в других приложениях.

Разделяемость:

Возможность работы со структурированными компонентами документа позволяет строить целый документ из составных частей, разбросанных по организации. Это позволяет пользователям делиться информацией без ее дублирования.

Чтобы начать работать с документами в SGML пользователю нужны два основных средства: редактор и средства экспорта (форматирования).

SGML-редактор

SGML-редактор отличается как от привычных текстовых редакторов, так и от "word processor"-ов. От первых - наличием поддержки структурированных документов, от вторых - отсутствием поддержки визуального форматирования. Редактор разбирает DTD редактируемого документа и "ведет" пользователя в соответствии с ним. Например, если DTD предусматривает элемент <SECTION>, в котором могут встречаться только элементы <SUBSECTION> или <PARAGRAPH>, то пользователю, редактирующему элемент <SECTION>, будет предложено вставить один из этих двух разрешенных элементов. SGML-редактор также обычно содержит средства навигации по иерархии документ.

Из числа популярных SGML-редакторов можно назвать ArborText ADEPT*Editor, SoftQuad Author/Editor, psgml, Adobe FrameMaker+SGML, Corel WordPerfect, и множество других.

Средства форматирования

Существует множество средств работы с SGML текстами. Большую их часть составляют средства форматирования - экспорта SGML в другие форматы для печати, просмотра и т.п. Выходные форматы могут быть любыми, завися лишь от доступного программного обеспечения и нужд пользователя. Например,

конвертеры в HTML, RTF и LATEX.

SGML-процессоры могут быть устроены по-разному. Существует несколько поколений таких средств (стоит вспомнить, что SGML отсчитывает уже второй десяток лет своей истории). Обычно они включают:

анализатор, разбирающий SGML документ, проверяющий корректность документа и строящий некоторое внутреннее представление иерархии элементов документа;

ядро, предоставляющее базовые функции работы с SGML (возможно, объединенное с анализатором в единую программу);

набор спецификаций, задающих ядру программы для конкретной обработки документа.

Синтаксический разбор SGML довольно сложен, поэтому полноценных анализаторов существует немного. Эталонным считается пакет SP.

Спецификации, или стили, пишутся на предлагаемом ядром языке программирования. Есть SGML-процессоры, программируемые на языках Perl, Tcl, диалектах Lisp, и т.п. Каждый процессор предлагает собственное представление иерархии документа и собственные примитивы работы с ним.

Такое положение призвано изменить принятый стандарт DSSSL (Document Style Semantics and Specification Language). Он специфицирует единый язык и интерфейсы SGML-процессоров. Используемый в нем язык программирования близок к популярному функциональному языку Scheme.

Используемые программы

В качестве SGML-редактора можно использовать XEmacs, включающий в комплект поставки SGML-модуль psgml. Ядром SGML-процессора выбрать CoST. Для форматирования в HTML, RTF и LATEX написан набор спецификаций на CoST. Использовать nsgmls, парсер из пакета SP. Управлению версиями помогает пакет контроля версий RCS.

Выбор DTD

При переходе к использованию технологии SGML встает вопрос выбора DTD. Обычно без проб и ошибок обойтись не удается. Можно пробовать "стандартные" DTD, широко используемые в индустрии, например, TEI Lite или DocBook. Можно

создавать свои DTD, ориентированные на типичные документы, встречающиеся в повседневной работе. Переход между DTD ввиду использования SGML-процессоров обычно безболезнен, поэтому тут возможно длительное экспериментирование.

SGML и Web

Язык форматирования Web-страниц HTML изначально вводился как приложение SGML. Позже, с бурным развитием WWW, HTML начал всячески расширяться с целью дать автору больший контроль над внешним представлением информации. Новые элементы и атрибуты, такие как <FONT> или <BGCOLOR>, ориентировались на визуальное форматирование. Появились и стали активно использоваться средства, не входящие собственно в язык разметки: imagemaps, Java и JavaScript, plugins, и прочее. Много появилось также элементов HTML, поддерживаемых только определенным браузером, или по-разному работающих в разных браузерах. Поэтому сейчас уже сложно утверждать, является ли HTML приложением SGML или нет. Очень немногие страницы создаются в соответствии со спецификациями на HTML и соответствующими DTD.

Эту проблему отчасти призваны облегчить каскадируемые стили, стандарт на которые принят W3 консорциумом. CSS1 отделяет стиль, задающий визуальное представление элементов, от разметки элементов.

Публикация в SGML

Если документы делаются доступными через WWW, то их придется переводить в формат HTML. Это можно делать заранее, а можно - "на лету", используя CGI или аналогичный интерфейс WWW сервера.

3.2. Основные части документа SGML

SGML-декларация — определяет, какие символы и ограничители могут появляться в приложении;

Document Type Definition — определяет синтаксис конструкций разметки. DTD может включать дополнительные определения, такие, как символьные ссылки-мнемоники;

Спецификация семантики, относится к разметке — также даёт ограничения синтаксиса, которые не могут быть выражены внутри DTD;

Содержимое SGML-документа — по крайней мере, должен быть корневой элемент.

SGML предоставляет множество вариантов синтаксической разметки для использования различными приложениями. Изменяя SGML-декларацию, можно даже отказаться от использования угловых скобок, хотя этот синтаксис считается стандартным, так называемым concrete reference syntax.

Пример синтаксиса SGML:

<QUOTE TYPE="example">

typically something like <ITALICS>this</ITALICS>

</QUOTE>

SGML стандартизован ISO: «ISO 8879:1986 Information processing—Text and office systems—Standard Generalized Markup Language (SGML)»

HTML и XML произошли от SGML. HTML — это приложение SGML, а XML — это подмножество SGML, разработанное для упрощения процесса машинного разбора документа. Другими приложениями SGML являются SGML Docbook (документирование) и «Z Format» (типография и документирование).

Заключение

В век цифровых технологий существует множество программ для создания web сайтов. Также, как и существуют различные языки разметки гипертекста.

В данной курсовой работе были рассмотрены такие языки разметки гипертекста, как SGML, XML, HTML. И выявлено, что HTML и XML произошли от SGML. HTML — это приложение SGML, а XML — это подмножество SGML, разработанное для упрощения процесса машинного разбора документа. XML - Extensible Markup Language, то есть Расширяемый Язык Разметки, возник в результате развития языка HTML (HyperText Markup Language, языка разметки гипертекста). Однако было бы грубой ошибкой воспринимать его лишь как некую усовершенствованную версию языка HTML. По сути, язык XML представляет собой новое поколение языков разметки.

Были рассмотрены структуры написания страниц посредством разных языков. Сходства и различия представленных вариантов. И выявлено, что XML является более развитым языком разметки по сравнению с HTML. HTML применяется главным

образом для представления содержания текстовых документов, а XML используется для структурирования данных.

XML — это не замена HTML. И это не просто HTML с дополнительными тегами, хотя описание XML гораздо больше по своему объему, нежели описание HTML. Несмотря на то, что HTML является приложением SGML, а XML является подмножеством SGML, HTML и XML имеют существенные различия. В частности, XML позволяет создавать собственные теги для различных целей. Кроме того, различия более ощутимы, если принять во внимание те проблемы, для решения которых применяется XML. Можно сказать, что XML представляется решением обычных проблем, вызванных применением HTML. При помощи XML достигается:

Лучший контроль над размещением информации.

Меньшая загрузка Web-сервера благодаря возможностям по доступу к информации на клиентской стороне.

Применение различных типов гиперссылок (hyperlinks).

Возможность распространения различных видов информации в Internet и intranet.

Меньшее количество проблем, возникающих при отображении больших страниц (long pages).

Тем не менее, несмотря на то что XML считается более «качественным» языком, HTML уверенно удерживает первое место по популярности.

Знания, полученные мною при написании данной курсовой работы, пригодятся мне в будущем на работе, мне следует ещё многое изучить, чтобы стать профессионалом в данном деле, но я буду и дальше изучать web-разработку на обозначенных языках и совершенствоваться как специалист.

Список использованной литературы:

1. Сухов К. HTML5 – путеводитель по технологии. – М.: ДМК Пресс, 2013. – 352 с.
2. Дженнифер Нидерст Роббинс HTML5, CSS3 и JavaScript. Исчерпывающее руководство. 4-ое издание 2014г, 516 с.
3. Робсон Э., Фримен Э. Изучаем HTML, XHTML и CSS. 2-е изд. — СПб.: Питер, 2014. — 720 с.
4. Олифер В. Г., Олифер Н.А. Компьютерные сети. СПб.: Питер, 2005 - 864 с.

5. Кох Дж., Дэвидсон К. XML. Огромные возможности и легкость изучения, 2007г, 256 с.
6. Учебник PHP - "Для Чайника" URL: http://project.net.ru/web-master/php/article1/ (Дата обращения 25.03.2019)
7. Что такое HTML и краткая история HTML URL: http://netagent.chat.ru/html/ul1.html (Дата обращения 15.03.19).
8. Иллюстрированный самоучитель по созданию сайтов URL: http://project.net.ru/others/article3/ (Дата обращения 15.04.2019)
9. Учебник по Проекту Документирования FreeBSD для новых участников URL: http://www.ntwk.ru/docs/freebsd/fdp-primer/sgml-primer.html (Дата обращения 12.04.2019)
10. Коротко об HTML 5.2 URL: https://habr.com/ru/post/345388/ (Дата обращения 28. 04. 2019)
11. World Wide Web Consortium URL: https://www.w3.org/TR/html52/changes.html#new-features (Дата обращения 06.04.2019)
12. Basic Web Page Building(Создание базовой веб-страницы) - URL: https://www.arachnoid.com/arachnophilia/Documentation/BasicWebPageBuilding.html (Дата обращения 24.04.19).