Школа

XML

Данная серия документов подготовлена на основе материалов сайта Школы Консорциума W3C. Этот сайт является экспериментальным сервером, на котором содержание документов хранится в формате XML. Пользователям сайта эти документы доступны в виде HTML (преобразование на строне клиента с помощью таблицы стилей XSLT) и в виде PDF (преобразование тех же документов в XSL-FO, а затем в формат PDF).

Добро пожаловать в школу XML

Школа XML[1] (open link)

В школе XML вы узнаете что такое XML и в чем разница между XML и HTML. Кроме того вы узнаете как начать использовать XML для решения своих задач. Изучайте XML!

Примеры XML[2] (open link)

Учитесь на примерах! С помощью нашего редактора вы сможете редактировать XML-документ и нажимая на тестовую кнопку сразу увидеть результат. Попробуйте!

Тест на знание XML[3] (open link)

Проверьте свои навыки в XML! Пройдите тест на знание XML!

Книги по XML[4] (open link)

Ищите хорошую книгу по XML? Почитайте наш обзор!

Основы XML: содержание

Введение в XML[5] (open link)

Что такое XML и чем он отличается от HTML?

Как можно использовать XML[6] (open link)

Несколько различных способов применения XML.

Синтаксис XML[7] (open link)

Простые и очень строгие правила XML.

Элементы XML[8] (open link)

Элементы XML, отношения между ними, содержание и правила наименования.

Атрибуты XML[9] (open link)

Как можно использовать атрибуты XML для описания элементов или для предоставления дополнительной информации о них.

Проверка правильности XML[10] (open link)

1: http://xml.nsu.ru/xml/xml_intro.xml
2: http://xml.nsu.ru/xml/xml_examples.xml
3: http://www.w3schools.com/xml/xml_quiz.asp
4: http://www.w3schools.com/xsl/xsl_books.asp
5: http://xml.nsu.ru/xml/xml_intro.xml
6: http://xml.nsu.ru/xml/xml_usedfor.xml
7: http://xml.nsu.ru/xml/xml_syntax.xml
8: http://xml.nsu.ru/xml/xml_elements.xml
9: http://xml.nsu.ru/xml/xml_attributes.xml
10: http://xml.nsu.ru/xml/xml_dtd.xml

Разница между правильным и пригодным XML-документом, как DTD применяется для определения XML-документа.

Поддержка XML в Netscape и Explorer[11] (open link)

О поддержке XML двумя самыми популярными браузерами.

Просмотр XML в Internet Explorer[12] (open link)

Как использовать Internet Explorer для просмотра XML-файла.

Демонстрация XML с помощью CSS[13] (open link)

Как использовать Internet Explorer и CSS для демонстрации XML-файла.

Демонстрация XML с помощью XSL[14] (open link)

Как использовать Internet Explorer и XSL для демонстрации XML-файла.

XML, встроенный в HTML[15] (open link)

Как встроить XML внутрь HTML-документов.

Парсер Microsoft XML[16] (open link)

Как применять парсер Microsoft XML для открытия и манипуляций с документами XML.

XML в реальной жизни[17] (open link)

Здесь мы поговорим о нескольких возможных применениях XML в реальной жизни.

Вторая ступень XML: содержание

Пространства имен XML[18] (open link)

Как избегать конфликтов между именами элементов используя пространства имен.

XML CDATA[19] (open link)

Как дать указание парсеру XML не анализировать текст.

Кодировка в XML[20] (open link)

Как задать кодировку ваших XML-документов.

Сервер XML[21] (open link)

Как генерировать XML на сервере.

XML-приложения[22] (open link)

Как применять IE5 для навигации по XML-файлу и как создать завершенное приложение на XML.

XML HTTP-запросы[23] (open link)

Как запрашивать у сервера XML используя HTTP.

Поведения для HTML и XML[24] (open link)

Как можно использовать новый CSS-селектор поведения для создания динамического кон-

11: http://xml.nsu.ru/xml/xml_browsers.xml
12: http://xml.nsu.ru/xml/xml_view.xml
13: http://xml.nsu.ru/xml/xml_display.xml
14: http://xml.nsu.ru/xml/xml_xsl.xml
15: http://xml.nsu.ru/xml/xml_data_island.xml
16: http://xml.nsu.ru/xml/xml_parser.xml
17: http://xml.nsu.ru/xml/xml_real_life.xml
18: http://xml.nsu.ru/xml/xml_namespaces.xml
19: http://xml.nsu.ru/xml/xml_cdata.xml
20: http://xml.nsu.ru/xml/xml_encoding.xml
21: http://xml.nsu.ru/xml/xml_server.xml
22: http://xml.nsu.ru/xml/xml_applications.xml
23: http://xml.nsu.ru/xml/xml_http.xml
24: http://xml.nsu.ru/xml/xml_behaviors.xml

тента.

Технологии XML[25] (open link)

Технологии XML, которые важны для понимания и разработки приложений на XML.

Примеры и тест по XML

Примеры XML[26] (open link)

Масса примеров XML!

Тест на знание XML[27] (open link)

Проверьте свои знания XML с помощью нашего теста!

Ресурсы по XML

Книги по XML[28] (open link)

Ищете хорошую книгу по XML? Почитайте наше обозрение!

Введение в XML

XML создан для описания данных и фокусируется на том, что именно они из себя представляют
HTML создан для демонстрации данных и фокусируется на том, как данные выглядят

Что вы должны уже знать

Прежде, чем вы продолжите изучение XML, вы должны уже иметь общее представление о следующих вещах:

- WWW, HTML и основы создания веб-страниц
- Скриптовые языки для Web, например, JavaScript или VBScript

Если вы хотите предварительно изучить эти вопросы, то прежде чем начать обучение в школе XML, найдите учебники по этим темам на главной странице W3Schools: [29] (open link)

Что такое XML?

- XML - это EXtensible Markup Language, в переводе "расширяемый язык разметки"
- XML является языком разметки, также как и HTML.
- XML был создан для описания данных.
- Тэги XML не заданы в этом языке с самого начала. Вы должны определить свои собственные тэги.
- Для описания данных XML использует DTD (Document Type Definition - Определение типа документа).

25: http://xml.nsu.ru/xml/xml_technologies.xml
26: http://xml.nsu.ru/xml/xml_examples.xml
27: http://www.w3schools.com/xml/xml_quiz.asp
28: http://www.w3schools.com/xsl/xsl_books.asp
29: http://www.w3schools.com/

- XML в совокупности с DTD сконструирован так чтобы быть само-описываемым.

Главное различие между XML и HTML

XML предназначен для хранения данных

XML не является заменой HTML.

XML и HTML создавались с различными целями:

XML создавался для описания данных и фокусируется на том, чем являются данные.

HTML создавался для демонстрации данных и фокусируется на том, как данные выглядят.

HTML предназначен для показа информации, а XML - для описания информации.

XML не делает ничего

XML не предназанчен для того, чтобы что-нибудь делать.

Может быть, это немного трудно понять, но XML ничего не делает. Он не для того был создан. XML был создан для структурирования, хранения и передачи информации.

Вот пример записки Бобу от Джани, сохраненная как XML:

```
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

У записки есть заголовок и содержательная часть (body). Кроме того, в ней содержится информация об отправителе (from) и получателе (to). Тем не менее этот XML-документ не делает ничего. Это просто чистая информация, упакованная в XML-тэги. Еще нужно написать какое-то программное обеспечение, чтобы отправить эту записку, получить ее или показать ее на экране.

XML - свободен и расширяем

Тэги XML не определены с самого начала. Вы должны "изобрести" свои собственные тэги.

Тэги, которые используются для того, чтобы разметить HTML-документ заданы изначально. Автор HTML-документа может использовать только те тэги, которые определены в стандарте HTML (например <p> или <h1...>).

XML позволяет автору определить свои собственные тэги и свою собственную структуру документа.

Тэги в приведенном примере (например, <to> или <from>), не заданы в стандарте XML. Эти тэги "изобретены" автором XML-документа.

XML дополняет HTML

XML не является заменой HTML.

Важно понимать, что XML не является заменой HTML. Скорее всего, в будущем XML будет применяться для описания данных, в то время как HTML будет применяться для форматирования и демонстрации этих данных.

Можно описать XML как независимый от платформы, от железа и от программного обеспечения инструмент для обмена информацией.

XML и будущее развитие Web

XML будет использоваться повсюду.

Мы принимаем участие в развитии языка XML с самого момента его создания. Удивительно видеть, насколько быстро были разработаны стандарты XML и как быстро множество разработчиков программного обеспечения их приняли.

Мы глубоко убеждены, что в будущем XML будет настолько же важен для Web, как и HTML был важен для создания Web, и что XML станет самым общепринятым инструментом для оперирования и обмена данными.

XML-шутка

Вопрос: когда мне следует применять XML?

Ответ: когда вы хотите видеть в своем резюме модное словечко.

Как можно использовать XML?

Важно понимать, что XML был создан для хранения, транспортировки и обмена данными. Он не предназначен для их отображения

XML может отделить данные от HTML

С помощью XML можно устроить хранение данных вне HTML-кода

Когда для отображения данных используется HTML, данные хранятся прямо в HTML-коде. Использование XML позволяет хранить данные в отдельных XML-файлах. Благодаря этому, вы можете сосредоточиться на применении HTML для отображения данных и быть уверенными в том, что изменения в данных не потребуют никакого изменения HTML-кода.

Кроме того, XML-данные можно хранить и в самих HTML-страницах в виде "островков данных". При этом вы по-прежнему сосредотачиваетесь только на применении HTML для форматирования и отображения данных.

XML применяется для обмена данными

С помощью XML можно устроить обмен данными между несовместимыми системами

В реальных условиях компьютерные системы и базы данных хранят данные в несовместимых форматах. Одной из самых сложных задач для разработчиков программного обеспечения было обеспечить обмен данными между такими системами через Интернет.

Преобразование данных в XML может в значительной мере уменьшить сложность этой задачи

и дать формат, который может быть прочитан многими типами приложений.

XML и технологии B2B

С помощью XML можно обмениваться через Интернет финансовой информацией

В ближайшем будущем ожидается возникновение многих B2B-технологий (Business To Business), основанных на XML.

XML постепенно становится главным языком обмена финансовой информацией между предприятиями через Интернет. В стадии разработки сейчас находится множество интересных B2B-приложений.

XML можно применять для организации совместной обработки данных

XML позволяет хранить совместно используемые данные в простых текстовых файлах

Поскольку XML хранит данные в текстовом формате, открывается возможность организации совместного использования данных, независимо от программного обеспечения или железа.

Становится гораздо легче создавать данные, с которыми могут работать различные приложения. Кроме того, гораздо легче расширять или проводить апгрейд системы со сменой операционной системы, сменой сервера, программного обеспечения или нового браузера.

XML можно использовать для хранения данных

XML позволяет хранить данные в текстовых файлах

Также XML может применяться для хранения данных в файлах или базах данных. Для извлечения данных из хранилища могут быть созданы специальные приложения, родственные приложения могут применяться и для отображения данных.

XML может сделать данные более полезными

XML позволяет сделать данные доступными для большего числа пользователей

Поскольку XML не зависит от железа и программного обеспечения, вы можете сделать свои данные доступными не только для стандартных HTML-браузеров.

К вашим источникам данных могут получить доступ и другие клиенты и приложения, почти также, как они получают доступ к базам данных. Можно сделать ваши данные доступными для всех видов "воспроизводящих машин" (агентов), становится гораздо легче обеспечить доступ к данным слепым людям и людям с другими отклонениями.

XML можно использовать для создания новых языков

XML - отец технологии WAP и языка WML

Безпроводной язык разметки WML (Wireless Markup Language) применяется для разметки интернет-приложений для носимых устройств, таких как мобильные телефоны. Он написан на XML.

Вы можете узнать больше о WML на нашей школе WML School:[30] (open link)

Если у разработчиков есть ум

Если он у них действительно есть, все будущие приложения будут хранить данные в формате XML

В будущем мы можем увидеть текстовые процессоры, электронные таблицы и базы данных, которые могут понимать данные других приложений, представленные в виде чистых текстовых файлов безо всяких специальных конверторов.

Остается только надеяться, что Microsoft и все другие разработчики программного обеспечения найдут согласие.

Синтаксис XML

Синтаксические правила XML очень просты и строги. Их легко запомнить и легко использовать
Поэтому разработка программного обеспечения для чтения и манипуляции с XML не трудоемка

Пример XML-документа

XML-документы используют само-описываемый и простой синтаксис

```
<?xml version="1.0"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

В первой строке этого документа - в XML-декларации - определяется версия XML, используемая в документе. В данном случае документ соответствует спецификации XML 1.0.

В следующей строке описывается корневой элемент документа (в ней как бы сообщается: "Этот документ - записка (note)"):

```
<note>
```

В следующих четырех строках описываются четыре дочерних элемента корневого элемента (to, from, heading, и body):

```
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
```

И, наконец, в последней строке задается конец корневого элемента:

```
</note>
```

Можете ли вы понять из этого документа, что в нем содержится записка (note), адресованная Tove от Jani? Разве не верно, что XML вполне обладает свойством само-описываемости?

30: http://www.w3schools.com/wap/default.asp

В XML все элементы должны иметь закрывающий тэг

В XML не разрешается опускать конечные тэги элементов

В HTML некоторые элементы могут не иметь закрывающего тэга. Следующий пример кода вполне приемлем в HTML:

```
<p>Это параграф
<p>Это еще один параграф
```

В XML все элементы обязательно должны иметь закрывающий тэг, вот так:

```
<p>Это параграф</p>
<p>Это еще один параграф</p>
```

Обратите внимание: вы, наверное, заметили, что XML-декларация в приведенном примере не имела закрывающего тэга. Это не ошибка. Дело в том, что эта декларация сама по себе не является частью XML-документа. Она не является XML-элементом и может не иметь закрывающего тэга.

В тэгах XML учитывается регистр

В отличие от HTML, в XML-тэгах учитывается регистр

В XML тэг <Letter> отличается от тэга <letter>.

Таким образом, начальные и конечные тэги должны писаться в одном регистре:

```
<Message>Вот так неправильно</message>
<message>Вот так правильно</message>
```

Элементы XML должны быть правильно вложены друг в друга

Неверное вложение элементов считается в XML неправильным

В HTML некоторые элементы могут быть вложены друг в друга неправильно, например:

```
<b><i>Этот текст пишется полужирным курсивом</b></i>
```

В XML все элементы должны быть правильно вложены друг в друга, например:

```
<b><i>Этот текст пишется полужирным курсивом</i></b>
```

XML-документы должны иметь единственный корневой элемент

Первый тэг XML-документа является корневым тэгом

Все XML-документы должны иметь единственную пару тэгов, задающую корневой элемент. Все остальные элементы должны быть вложены в корневой элемент.

Все элементы могут иметь под-элементы (дочерние элементы). Под-элементы должны быть правильно вложены внутри родительского элемента.

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
```

```
</root>
```

Значения атрибутов всегда должны быть заключены в кавычки

В XML не разрешается опускать кавычки вокруг значения атрибутов

В XML элементы могут обладать атрибутами в парах имя атрибута/его значение, также, как HTML. Но в XML значения атрибутов всегда должны быть заключены в кавычки. Исследуйте два приведенных ниже XML-документа. Первый из них - неправильный, второй - правильный:

```
<?xml version="1.0"?>
<note date=12/11/99>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

```
<?xml version="1.0"?>
<note  date="12/11/99">
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Ошибка в первом документе - атрибут date в элементе note не заключен в кавычки. Вот так правильно: date="12/11/99", а вот так - не правильно: date=12/11/99.

В XML пробелы сохраняются

В XML пробелы в вашем документе не будут устраняться

Этим XML отличается от HTML. В HTML предложение:

```
Привет,              меня зовут Толя
```

будет отображаться так:

```
Привет, меня зовут Толя
```

потому что в HTML лишние пробелы устраняются.

В XML, символы CR / LF преобразуются в LF

В XML перевод строки всегда делается с помощью символа LF

Знаете ли вы, что такое печатная машинка? Это такое механическое устройство, оно использовалось в прошлые века :-)

После того, как вы заканчивали печатать на ней строку текста, вам нужно было вручную вернуть печатную каретку к началу страницы и вручную же сдвинуть бумагу на одну строку вверх.

В приложениях под Windows новая строка текста обычно отмечается парой символов CR LF (carriage return, line feed - возврат каретки, перевод строки). В приложениях под Unix новая

строка обычно отмечается символом LF. В некотрых приложениях для отметки новой строки используется только символ CR.

В XML нет ничего особенного

В XML нет ничего особенного, это просто текст с добавленными XML-тэгами, заключенными в угловые скобки

Программное обеспечение, которое работает с простым текстом сможет работать и с XML. XML-тэги будут видны в любом текстовом редакторе и с ними не нужно работать как-то особенно.

В распознающих XML приложениях, однако, XML-тэги могут обрабатываться особым образом. Они могут быть невидимыми или иметь какое-то функциональное значение, в зависимости от природы данного приложения.

Элементы XML

Элементы XML расширяемы, они обладают определенными отношениями
Элементы XML подчиняются простым правилам наименования

Элементы XML расширяемы

Элементы XML можно расширять для того, чтобы они могли включить в себя больше информации

Взгляните на следующий пример XML-документа NOTE:

```
<note>
<to>Tove</to>
<from>Jani</from>
<body>Don't forget me this weekend!</body>
</note>
```

Теперь представим, что мы создали приложение, которое извлекает элементы <to>, <from> и <body> из XML-документа и выдает следующий результат:

```
MESSAGE
To: Tove
From: Jani

Don't forget me this weekend!
```

Пусть теперь автор XML-документа добавил в него дополнительную информацию:

```
<note>
<date>1999-08-01</date>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Нарушится ли в этом случае работа нашего приложения?

Нет. Приложение по-прежнему сможет извлечь из XML-документа элементы <to>, <from> и

<body> и выдать некоторый результат.

XML-элементы расширяемы.

XML-элементы обладают определенными отношениями

Элементы XML находятся во взаимных родительских и дочерних отношениях

Чтобы понять терминологию XML, вам нужно знать как называются отношения между XML-элементами и как описывается содержание элементов.

Пусть это - описание книги:

```
Book Title: Начала XML

Chapter 1: Введение в XML

Что такое HTML
Что такое XML

Chapter 2: Синтаксис XML

Элементы должны иметь конечный тэг
Элементы должны быть правильно вложены
```

А вот это пусть будет XML-документом, описывающем книгу:

```
<book>
<title>Начала XML</title>
<prod id="33-657" media="paper"></prod>
<chapter>Введение в XML
<para>Что такое HTML</para>
<para>Что такое XML</para>
</chapter>

<chapter>Синтаксис XML
<para>Элементы должны иметь конечный тэг</para>
<para>Элементы должны быть правильно вложены</para>
</chapter>

</book>
```

Book (книга) - корневой элемент. Title (заглавие) and chapter (глава) - дочерние элементы элемента book. Элемент book - родительский элемент элементов title и chapter. Элементы title и chapter соотносятся друг с другом как "соседи", поскольку они имеют один и тот же родительский элемент.

Элементы XML обладают определенным содержимым

Элементы XML могут обладать содержимым различного типа

XML-элементом является все, что заключено между начальным тэгом элемента и конечным тэгом элемента, включая и сами тэги.

Элемент может иметь элементное содержимое, смешанное содержимое, простое содержимое или он может быть пустым. Элементы также могут иметь атрибуты.

В выше приведенном примере элемент book имеет элементное содержимое, поскольку он содержит другие элементы. Элемент chapter имеет смешанное содержимое, потому что он содержит и текст и другие элементы. Элемент para имеет простое содержимое (или текстовое содержимое), поскольку он содержит только текст. Элемент prod - пустой, поскольку не содержит информации.

В этом примере только элемент prod имеет атрибуты. Атрибут, который называется id, имеет значение "33-657". Атрибут с названием media имеет значение "paper".

Наименование элементов

Названия XML-элементов должны подчиняться следующим правилам:

- Названия могут содержать буквы, цифры и другие символы
- Названия не могут начинаться с цифры или знака препинания
- Названия не могут начинаться с букв xml (или XML или Xml ..)
- В названии не должно быть пробелов

Когда вы станете "изобретать" названия элементов, постарайтесь придерживаться этих простых правил:

Можно использовать любые слова, никакие слова не являются зарезервированными, но хорошо, чтобы название было осмысленным. Имена с символом подчеркивания вполне годятся.

Пример: <first_name>, <last_name>.

Избегайте в названиях символов "-" и ".". Может произойти ошибка, когда ваше программное обеспечение попытается извлечь "name" из "first" (в элементе first-name) или решит, что "name" - это свойство объекта "first" (в элементе first.name).

Названия элементов могут быть какой угодно длины, но не злоупотребляйте. Имена должны быть короткими и простыми, например, <book_title>, а не <the_title_of_the_book>.

XML-документы часто связаны с некоторой базой данных, в которой существуют поля, соответствующие элементам XML-документа. Хорошим правилом может быть наименование элементов XML-документа по названиям соответствующих полей базы данных.

Не-английские символы вполне приемлемы в названиях элементов, но следите за тем, чтобы не возникло проблем, если программное обеспечение не поддерживает не-английские символы.

Символ ":" не следует использовать в названиях элементов, поскольку он зарезервирован для использования в так называемых пространствах имен (см. ниже).

Атрибуты XML

Элементы XML могут содержать в начальном тэге атрибуты, также, как в HTML
Атрибуты применяются для предоставления дополнительной информации об элементах

Атрибуты XML

XML-элементы могут иметь атрибуты

В HTML вы встречали похожее выражение: <IMG SRC="computer.gif">. Атрибут SRC дает дополнительную информацию об элементе IMG.

И в HTML и в XML атрибуты предоставляют об элементах дополнительную информацию:

```
<img src="computer.gif">
<a href="demo.asp">
```

Атрибуты часто несут информацию, которая не является частью основных данных. В приведенном ниже примере тип файла (type) не имеет прямого отношения к данным, но важен для программного обеспечения, которое производит манипуляции с этим элементом:

```
<file type="gif">computer.gif</file>
```

Стиль кавычек: двойные или одинарные?

Значения атрибутов всегда должны быть заключены в кавычки, хотя можно использовать и одинарные и двойные. Например, при указании пола в атрибуте sex, можно записать его так:

```
<person sex="female">
```

Или так:

```
<person sex='female'>
```

Двойные кавычки чаще используются, но иногда (когда значение атрибута само содержит кавычки) необходимо применять одинарные, например здесь:

```
<gangster name='George "Shotgun" Ziegler'>
```

Применять элементы или атрибуты?

Данные можно сохранять как в дочерних элементах, так и в атрибутах элемента

Взгляните на эти примеры:

```
<person sex="female">
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>

<person>
  <sex>female</sex>
  <firstname>Anna</firstname>
  <lastname>Smith</lastname>
</person>
```

В первом примере sex - атрибут. Во втором примере sex - дочерний элемент. В обоих документах содержится одна и та же информация.

Не существует правил, указывающих, когда использовать атрибуты, а когда - дочерние элементы. Можно сказать следующее: атрибуты широко применяются в HTML, но в XML не следует увлекаться ими. Для хранения важной информации лучше пользоваться дочерними элементами.

Как я поступаю

Мне нравится хранить данные в дочерних элементах

Следующие три XML-документа содержат одну и ту же информацию:

В первом примере используется атрибут date:

```
<note date="12/11/99">
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Во втором примере используется элемент date:

```
<note>
<date>12/11/99</date>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

В третьем примере (он мне нравится больше всего) применяется расширенный элемент date:

```
<note>
<date>
  <day>12</day>
  <month>11</month>
  <year>99</year>
</date>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Избегать использовать атрибуты?

Нужно ли избегать использования атрибутов?

Вот некоторые проблемы, возникающие при использовании атриутов:

- атрибуты не могут содержать множественные значения (дочерние элементы - могут)
- атрибуты трудно расширять (при последующих изменениях)
- атрибуты не описывают структуру (дочерние элементы - могут)
- атрибуты труднее обрабатывать программному обеспечению
- значения атрибутов трудно проверять с помощью DTD

Если вы применяете атрибуты для хранения данных, в конце концев вы можете получить документ, который трудно читать и обрабатывать. Старайтесь для описания данных использовать элементы. Применяйте атрибуты только для дополнительной информации, которая не имеет прямого отношения к данным документа.

Не доводите дело до такого положения (если вы думаете, что это похоже на XML, вы не понимаете сути):

```
<note day="12" month="11" year="99"
to="Tove" from="Jani" heading="Reminder"
body="Don't forget me this weekend!">
</note>
```

Исключения в моем правиле

В любом правиле всегда есть исключения

Мое правило применения атрибутов имеет одно исключение:

Иногда я добавляю к элементам атрибут-идентификатор ID. Это позволяет обращаться к XML-элементам точно также, как это делается с помощью атрибутов NAME или ID в HTML. Следующие примеры это демонстрируют:

```
<messages>
  <note ID="501">
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
  </note>

  <note ID="502">
    <to>Jani</to>
    <from>Tove</from>
    <heading>Re: Reminder</heading>
    <body>I will not!</body>
  </note>
</messages>
```

ID в этих примерах служит указателем, уникальным идентификатором, который позволяет различить отдельные элементы note в XML-файле, но не отдельные части этих элементов.

В общем: мета-данные (данные ОБ основных данных документа) следует хранить в атрибутах, а сами основные данные следует хранить в элементах.

Проверка правильности XML

XML-документ, имеющий правильный синтаксис, называется правильным (well formed) XML-документом
XML-документ, провереный с помощью DTD, называется пригодным (valid) XML-документом

"Правильные" XML-документы

"Правильные" XML-документы соответствуют правилам синтаксиса XML

"Правильный" XML-документ - это документ, который вполне подчиняется синтаксическим правилам XML, которые были описаны в предыдущих разделах:

```
<?xml version="1.0"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

"Пригодные" XML-документы

"Пригодный" XML-документ также соответствует и DTD

"Пригодный" XML-документ - это "правильный" XML-документ, который подчиняется правилам определения типа документа DTD (Document Type Definition):

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "InternalNote.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

XML DTD

В DTD задаются разрешенные элементы XML-документа

DTD предназначается для задания разрешенных строительных блоков XML-документа. В нем структура документа определяется как список разрешенных элементов. Вы можете узнать больше о DTD и о том, как с его помощью проверять правильность ваших XML-документов на нашей школе DTD.[31] (open link)

Схемы XML

XSchema - это основанная на XML альтернатива DTD

W3C поддерживает альтернативу DTD, которая называется схемы XML (XML Schema). Вы можете узнать больше о схемах XML на нашей школе XML-схем.[32] (open link)

Ошибки остановят вас

Ошибки в XML-документах вызовут остановку XML-программ

XML-спецификация консорциума W3C указывает, что программа не должна продолжать обработку XML-документа, если она обнаружит ошибку. Это сделано для того, чтобы было легко создавать программное обеспечение для XML и для того, чтобы все XML-документы были совместимы.

В HTML вполне возможно использовать документы, в которых имеется множество ошибок (например, зыбытый конечный тэг). Одна из главных причин, почему HTML-браузеры такие большие и не совместимы друг с другом состоит в том, что каждый браузер по-своему выясняет, как должен выглядеть документ, если браузер сталкивается с HTML-ошибками.

В XML такая ситуация невозможна.

31: http://xml.nsu.ru/dtd/dtd_home.xml
32: http://xml.nsu.ru/schema/schema_home.xml

Поддержка XML в Netscape и Explorer

На этом сайте мы сосредотачиваем внимание на поддержке XML, реализованной в Internet Explorer 5.0, потому что Netscape плохо поддерживает XML. Возможно, ситуация со временем изменится.

XML на этом сайте

Многие приложения поддерживают XML. Мы сосредоточим внимание на Internet Explorer 5.0

Некоторым это не понравится, но это сделано потому, что пока доступен лишь один простой способ продемонстрировать реальные примеры использования XML в Интернете.

Печально, но большая часть примеров будет действовать пока только в IE5 (мы ждем Netscape). Если вы хотите изучить XML без особых сложностей - на многих реальных примерах - придется с этим смириться.

XML в браузере Netscape Navigator

Netscape обещает реализовать полную поддержку XML в следующих версиях своего браузера

Мы надеемся, что Netscape включит в следующие версии своего браузера полную поддержку XML по рекомендациям W3C.

Основываясь на предыдущем опыте, мы можем только надеяться, что XML, реализованный в Navigator и Explorer будет совместим.

А пока, если вы хотите применять XML независимо от браузера, вам следует работать с XML на сервере и перед отправкой страниц пользователю преобразовывать его в HTML.

Вы можете узнать больше о преобразовании XML в HTML на нашей школе XSL School:[33] (open link)

XML в Internet Explorer 5.0

Internet Explorer 5.0 поддерживает стандарт XML 1.0

Браузер Internet Explorer 5.0 поддерживает большую часть международных стандартов XML 1.0 и XML DOM (Document Object Model). Эти стандарты установлены консорциумом World Wide Web Consortium (W3C).

Internet Explorer 5.0 поддерживает следующие XML-технологии:

- Просмотр XML-документов
- Полная поддержка W3C-стандартов по DTD
- XML, встроенный в HTML в виде островков данных
- Привязка XML-данных к HTML-элементам
- Трансформация и демонстрация XML с помощью XSL
- Демонстрация XML с помощью CSS
- Доступ к модели XML DOM

33: http://xml.nsu.ru/xsl/xsl_home.xml

Internet Explorer 5.0 также поддерживает технологию Behaviors (поведения):

- Behaviors - это используемая только Microsoft технология
- Технология Behaviors может отделить скрипты от HTML-страницы
- С ее помощью можно сохранять XML-данные на диске клиента

На этом сайте даны примеры всех этих возможностей.

Вы можете больше узнать об Internet Explorer и скачать его последнюю версию в разделе школ W3Schools Browser Information:[34] (open link)

Просмотр XML в Internet Explorer 5.0

Чистые XML-файлы можно просматривать в Internet Explorer 5.0, но чтобы они выглядели как нормальные веб-страницы, нужно добавить информацию об отображении
Чтобы познакомиться с приводимыми здесь примерами XML-файлов, вам нужен Internet Explorer 5.0 или более поздняя его версия

Просмотр XML в Internet Explorer 5.0

IE 5.0 можно использовать для просмотра любого XML-документа

Чтобы просмотреть XML-документ, кликните по ссылке, впишите URL в адресную строку, или два раза кликните на названии XML-файла в папке - все как обычно.

Когда вы откроете XML-документ в IE, цветом будут выделены XML-тэги документа. Если кликнуть на значок (+) или (-) слева от элемента, это вызовет свертывание или развертывание структуры этого элемента.

Если вы хотите взглянуть на чистый XML-исходник, выберете "View Source" в меню браузера.

Не ждите, что XML-файл будет отформатирован, как HTML-документ!

Кликните на иконке и IE откроет XML-документ note.xml:[35] (open xml)

Просмотр неверных XML-файлов

XML-файлы, содержащие ошибку, в IE открываются с сообщением об ошибке

Кликните на иконке и IE откроет неверный XML-документ note_error.xml, выдав сообщение об ошибке:[36] (open xml)

Другие примеры

Просмотр некоторых XML-документов поможет вам лучше понять XML

Для этого здесь собраны некоторые XML-файлы:

Каталог компакт-дисков на XML: это коллекция дисков моего отца, описанная на XML (старые и скучные названия, мне кажется):[37] (open xml)

Каталог растений на XML: это каталог растений цветочного магазина, написанный на

34: http://www.w3schools.com/browsers/default.asp
35: http://xml.nsu.ru/xml/note.xml
36: http://xml.nsu.ru/xml/note_error.xml
37: http://xml.nsu.ru/xml/cd_catalog.xml

[38] (open xml)

Простое меню: это меню завтрака в ресторане, написанное на XML:[39] (open xml)

Почему XML отображается таким образом?

XML-документы не содержат информации о том, как отображать содержащиеся в них данные

Поскольку тэги XML придумываются или "изобретаются" автором XML-документа, мы не знаем заранее, является ли <table> HTML-описанием таблицы или этот тэг описывает деревянный кухонный стол.

Безо всякой информации об отображении данных, большинство браузеров отобразят XML-документ таким, какой он есть.

В следующих разделах мы познакомимся с различными подходами при решении проблемы отображения: с применением CSS, XSL, JavaScript и островков данных.

Демонстрация XML с помощью CSS

С помощью каскадных таблиц стилей CSS (Cascading Style Sheets) к XML-документу можно добавить информацию об отображении

Отображать XML-файлы с помощью CSS?

Будет ли в будущем для форматирования XML-файлов применяться CSS?

Мы так не думаем. Но не будем возражать, если вы попробуете этот способ:

Вот чистый XML-файл - Каталог CD:[40] (open xml)

А вот его каскадная таблица стилей - файл CSS:[41] (open css)

И, наконец, вот каталог CD, отформатированный CSS-файлом:[42] (open xml)

Вот кусочек XML-файла, в котором содержится ссылка на таблицу стилей CSS:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="cd_catalog.css"?>
<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>
  <CD>
```

38: http://xml.nsu.ru/xml/plant_catalog.xml
39: http://xml.nsu.ru/xml/simple.xml
40: http://xml.nsu.ru/xml/cd_catalog.xml
41: http://xml.nsu.ru/xml/cd_catalog.txt
42: http://xml.nsu.ru/xml/cd_catalog_with_css.xml

```
    <TITLE>Hide your heart</TITLE>
    <ARTIST>Bonnie Tyler</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>CBS Records</COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1988</YEAR>
  </CD>
.
.
.
.
</CATALOG>
```

Мы не считаем, что применение XML совместно с CSS станет в будущем широко распространено. Даже если кажется, что тут можно применять CSS, мы уверены, что стандартом станет форматирование с помощью XSL (когда основные браузеры начнут его поддерживать).

Писать страницы на XML?

Будете ли вы в будущем писать веб-страницы на XML?

Мы так не думаем. Но не будем возражать, если вы попробуете: вот пример домашней странички, написанной на XML:[43] (open xml)

Мы не думаем, что XML будет использоваться для создания домашних страничек.

Мы считаем, что для этого будет использоваться XHTML - HTML, определенный как XML. Изучайте XHTML на наших школах W3Schools![44] (open link)

Демонстрация XML с помощью XSL

С помощью XSL к XML-документу можно добавить информацию об его отображении

Отображение XML с помощью XSL

XSL - наиболее предпочтительный язык стилей для XML

XSL - расширяемый язык стилей (the eXtensible Stylesheet Language) гораздо более богат возможностями, чем CSS. Один из способов применять XSL - это трансформировать XML в HTML перед тем, как документ будет отображен браузером. Следующие примеры демонстрируют этот метод:

Вот "сырой" XML-документ:[45] (open xml)

Это тот же самый документ, но здесь с помощью таблицы стилей XSL заданы параметры отображения stylesheet:[46] (open xml)

А вот сама таблица стилей XSL:[47] (open xsl)

43: http://xml.nsu.ru/xml/home.xml
44: http://www.w3schools.com/xhtml/default.asp
45: http://xml.nsu.ru/xml/simple.xml
46: http://xml.nsu.ru/xml/simplexsl.xml
47: http://xml.nsu.ru/xml/simple.xsl

Сокращенная копия XML-документа приведена ниже. Обратите внимание на ссылку на таблицу стилей, расположенную во второй строке:

```
<?xml version="1.0"?>
<?xml:stylesheet type="text/xsl" href="simple.xsl" ?>
<breakfast_menu>
  <food>
    <name>Belgian Waffles</name>
    <price>$5.95</price>
    <description>
       two of our famous Belgian Waffles
    </description>
    <calories>650</calories>
  </food>
</breakfast_menu>
```

Вы можете узнать больше о XSL, посетив нашу школу W3Schools:[48] (open link)

XML, встроенный в HTML

При использовании Internet Explorer 5.0, XML можно встраивать в HTML-страницы с помощью островков данных (data islands)

XML, встроенный в HTML

Для встраивания XML-данных в HTML-страницы применяется тэг <xml>, не являющийся частью HTML-стандарта

XML-данные можно встраивать прямо в HTML-страницы вот так:

```
<xml id="note">
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
</xml>
```

Можно также встроить отдельный XML-файл, вот так:

```
<xml id="note" src="note.xml">
</xml>
```

Обратите внимание, что тэг <xml> - это HTML-элемент, а не XML-элемент.

Привязка данных

Островки данных можно привязывать к определенным HTML-элементам (например, HTML-таблицам)

В приведенном ниже примере островок XML-данных, имеющий атрибут-идентификатор ID "cdcat" загружается из внешнего XML-файла. HTML-таблица привязывается к этому островку данных посредством атрибута datasrc (data source - источник данных) и, наконец, элементы

48: http://xml.nsu.ru/xsl/xsl_home.xml

таблицы привязываются к XML-данным с помощью атрибута datafld (data field - поле данных) элементов span.

```
<html>
<body>

<xml id="cdcat" src="cd_catalog.xml"></xml>

<table border="1" datasrc="#cdcat">
<tr>
<td><span datafld="ARTIST"></span></td>
<td><span datafld="TITLE"></span></td>
</tr>
</table>

</body>
</html>
```

Если вы используете IE 5, вы можете испытать эту схему в действии:[49] (open editor)

С помощью IE 5.0 вы также можете взглянуть на внешний XML-файл:[50] (open xml)

А в этом примере используются элементы <thead>, <tbody>, и <tfoot>:[51] (open editor)

Парсер Microsoft XML

Чтобы читать и модифицировать XML-документ, нужен XML-парсер

Использование XML-парсера

XML-парсер Microsoft идет в комплекте поставки Microsoft Internet Explorer 5.0

После установки IE 5.0 парсер становится доступным для вызова скриптами - как расположенными внутри HTML-документов, так и находящимися в ASP-файлах. Этот парсер основан на не зависимой от языка программной модели, которая поддерживает:

- JavaScript, VBScript, Perl, VB, Java, C++ и другие языки
- Спецификацию W3C XML 1.0 и XML DOM
- DTD и проверку правильности (валидацию) документа

Если вы используете с IE 5.0 JavaScript, вы можете создать объект XML-документ с помощью следующего кода:

```
var xmlDoc = new ActiveXObject("Microsoft.XMLDOM")
```

Если вы используете VBScript, вы можете создать объект XML-документ следующим кодом:

```
set xmlDoc = CreateObject("Microsoft.XMLDOM")
```

Если вы применяете VBScript в активных серверных страницах ASP (Active Server Page), можно использовать следующий код:

```
set xmlDoc = Server.CreateObject("Microsoft.XMLDOM")
```

49: http://www.w3schools.com/xml/tryit.asp?filename=cd_catalog_island
50: http://xml.nsu.ru/xml/cd_catalog.xml
51: http://www.w3schools.com/xml/tryit.asp?filename=cd_catalog_island_thead

Загрузка XML-файла в парсер

Используя скрипты, можно загружать XML-файлы в парсер

Следующий код загружает XML-документ (note.xml) в XML-парсер:

```
<script type="text/javascript">
var xmlDoc = new ActiveXObject("Microsoft.XMLDOM")
xmlDoc.async="false"
xmlDoc.load("note.xml")
// ....... здесь документ обрабатывается
</script>
```

В первой строке этого скрипта создается экземпляр XML-парсера Microsoft.

Во второй строке отключается асинхронная загрузка, чтобы парсер не начинал выполнение каких-либо операций, пока документ полностью не загрузится.

В третьей строке парсеру дается указание загрузить XML-документ, имеющий название note.xml.

Загрузка в парсер чистого XML-текста

XML-текст можно также загрузить посредством загрузки отдельных строк текста

Этот код загружает в XML-парсер строку текста:

```
<script type="text/javascript">

var text="<note>"
text=text+"<to>Tove</to><from>Jani</from>"
text=text+"<heading>Reminder</heading>"
text=text+"<body>Don't forget me this weekend!</body>"
text=text+"</note>"

var xmlDoc = new ActiveXObject("Microsoft.XMLDOM")
xmlDoc.async="false"
xmlDoc.loadXML(text)
// ....... processing the document goes here
</script>
```

Обратите внимание, что для загрузки строки текста используется метод "loadXML" (а не метод "load")

Отображение XML с помощью JavaScript

Для отображеня XML можно применять JavaScript

Для импорта данных из XML-файла и отображения XML-данных внутри HTML-документа можно использовать JavaScript (или VBScript).

Посмотрите, как XML и HTML могут взаимодополнять друг друга таким образом: взгляните сначала на XML-документ (note.xml):[52] (open xml)

Теперь откройте HTML-документ (note.htm), в котром содержится JavaScript, который читает

52: http://xml.nsu.ru/xml/note.xml

XML-файл и отображает эту информацию внутри заранее заданных элементов HTML-страницы:[53] (open htm)

Попробуйте этот механизм в действии:[54] (open editor)

Вы можете узнать больше о таком применении JavaScript на нашей школе DOM:[55] (open link)

XML в реальной жизни

Некоторые реальные примеры того, как можно использовать XML для хранения и обмена информацией

Пример: XML News

XMLNews - это спецификация для обмена новостями и другой информацией

Применение таких стандартов облегчает и производителям и потребителям новостей производить, получать и архивировать любые виды новостной информации не зависимо от железа, программного обеспечения и языков программирования.

Пример документа, оформленного с помощью XML News

```
<?xml version="1.0"?>

<nitf>

<head>
<title>Colombia Earthquake</title>
</head>

<body>

<body.head>
<headline>
<hl1>143 Dead in Colombia Earthquake</hl1>
</headline>
<byline>
<bytag>By Jared Kotler, Associated Press Writer</bytag>
</byline>
<dateline>
<location>Bogota, Colombia</location>
<story.date>Monday January 25 1999 7:28 ET</story.date>
</dateline>
</body.head>

</body>

</nitf>
```

53: http://xml.nsu.ru/xml/note.htm
54: http://www.w3schools.com/xml/tryit.asp?filename=note
55: http://xml.nsu.ru/dom/dom_home.xml

Дополнительную информацию о XMLNews можно найти здесь: http://www.xmlnews.org/:[56] (open link)

Пространства имен XML

Пространства имен XML дают метод избегать конфликты имен элементов

Конфликты имен

Поскольку имена элементов в XML не фиксированы, часто случаются конфликты, когда два различных документа используют одно и то же имя для описания двух различных типов элементов.

В этом XML-документе информация заключена в таблицу (table):

```
<table>
   <tr>
   <td>Apples</td>
   <td>Bananas</td>
   </tr>
</table>
```

А в этом XML-документе содержится информация о столе как части мебели (table):

```
<table>
   <name>African Coffee Table</name>
   <width>80</width>
   <length>120</length>
</table>
```

Если эти два XML-документа сложить вместе, возникнет конфликт имен элементов, потому что оба документа содержат элементы <table> с разным содержанием и определением.

Разрешение конфликтов имен применением префикса

В этом XML-документе информация заключена в таблицу (table):

```
<h:table>
   <h:tr>
   <h:td>Apples</h:td>
   <h:td>Bananas</h:td>
   </h:tr>
</h:table>
```

А в этом XML-документе содержится информация о столе как части мебели (table):

```
<f:table>
   <f:name>African Coffee Table</f:name>
   <f:width>80</f:width>
   <f:length>120</f:length>
</f:table>
```

Теперь конфликт имен преодолен, поскольку документы используют различные имена для элементов <table> (<h:table> и <f:table>).

56: http://www.xmlnews.org/

Используя префикс, мы создаем два различных типа элемента <table>.

Применение пространств имен (namespaces)

В этом XML-документе информация заключена в таблицу (table):

```
<h:table xmlns:h="http://www.w3.org/TR/html4/">
   <h:tr>
   <h:td>Apples</h:td>
   <h:td>Bananas</h:td>
   </h:tr>
</h:table>
```

А в этом XML-документе содержится информация о столе как части мебели (table):

```
<f:table xmlns:f="http://www.w3schools.com/furniture">
   <f:name>African Coffee Table</f:name>
   <f:width>80</f:width>
   <f:length>120</f:length>
</f:table>
```

Кроме применения самих префиксов в этих примерах к тэгу <table> был добавлен атрибут xmlns. Это сделано для того, чтобы дать префиксу квалификационное имя, связанное с так называемым пространством имен.

Атрибут namespace

Атрибут namespace помещается в начальный тэг элемента с использованием следующего синтаксиса:

```
xmlns:namespace-prefix="namespace"
```

В выше приведенных примерах само пространство имен было определено с использованием Интернет-адреса:

```
xmlns:f="http://www.w3schools.com/furniture"
```

В предложенном консорциумом W3C стандарте по пространствам имен говорится, что пространство имен само может быть универсальным идентификатором ресурса URI (Uniform Resource Identifier).

Когда пространство имен задается в начальном тэге элемента, все дочерние его элементы, обладающие тем же префиксом связываются с тем же пространством имен.

Обратите внимание, что адрес, используемый для идентификации пространства имен не задействуется парсером для поиска информации. Единственная задача - дать пространству имен уникальное имя. Тем не менее, очень часто компании используют пространство имен как указатель на реальную веб-страницу, содержащую информацию об этом пространстве имен. Попробуйте зайти на http://www.w3.org/TR/html4/:[57] (open link)

Универсальный идентификатор ресурса (URI)

Универсальный идентификатор ресурса (URI) - это строка символов, которая идентифицирует какой-то ресурс в Интернете. Наиболее распространенным типом URI является универсальный локатор ресурса URL (Uniform Resource Locator), который идентифицирует располо-

57: http://www.w3.org/TR/html4/

женный в Интернете доменный адрес. Другим, менее распространенным типом URI является универсальное имя ресурса URN (Universal Resource Name). В наших примерах мы используем только URL.

Поскольку в нашем мебельном примере для идентификации пространства имен используется интернетовский адрес, мы можем быть уверены, что наше пространство имен уникально.

Пространство имен, используемое по умолчанию

Задание пространства имен, используемого по умолчанию позволяет нам избежать необходимости указывать перфикс во всех дочерних элементах. Это делается с помощью следующего синтаксиса:

```
<element xmlns="namespace">
```

В этом XML-документе информация заключена в таблицу (table):

```
<table xmlns="http://www.w3.org/TR/html4/">
   <tr>
   <td>Apples</td>
   <td>Bananas</td>
   </tr>
</table>
```

А в этом XML-документе содержится информация о столе как части мебели (table):

```
<table xmlns="http://www.w3schools.com/furniture">
   <name>African Coffee Table</name>
   <width>80</width>
   <length>120</length>
</table>
```

Пространства имен в реальной жизни

Когда вы начнете применять XSL, вы увидите реальное использование пространств имен. Таблицы стилей XSL используются для преобразования XML-документов в другие форматы, например, HTML.

Если вы внимательно приглядитесь к приведенному ниже XSL-документу, вы увидите, что большая часть тэгов являются в нем тэгами HTML. Остальные тэги имеют префикс xsl, идентифицирующий пространство имен "http://www.w3.org/TR/xsl"[1]:

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/xsl">
<xsl:template match="/">
  <html>
  <body>
    <table border="2" bgcolor="yellow">
      <tr>
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="CATALOG/CD">
      <tr>
        <td><xsl:value-of select="TITLE"/></td>
```

1: http://www.w3.org/TR/xsl

```
        <td><xsl:value-of select="ARTIST"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

XML CDATA

Весь текст XML-документа анализируется парсером
Парсер игнорирует только текст, расположенный в разделе CDATA

Анализируемые данные

Обычно парсер анализирует весь текст XML-документа

Когда XML-элемент анализируется парсером, текст внутри тэгов XML также анализируется:

```
<message>Этот текст тоже анализируется</message>
```

Парсер это делает, потому что XML-элементы могут содержать другие элементы, например, в этом примере элемент<name> содержит два других элемента (first и last):

```
<name><first>Bill</first><last>Gates</last></name>
```

Парсер разбивает эту строку на под-элементы, примерно так:

```
<name>
   <first>Bill</first>
   <last>Gates</last>
</name>
```

Особые символы

Некоторые особые XML-символы должны заменяться ссылками на сущности

Если вы поместите внутрь XML-элемента символ "<", это вызовет ошибку, потому что парсер проинтерпретирует его как начало нового элемента. Поэтому писать вот так нельзя:

```
<message>if salary < 1000 then</message>
```

Чтобы избежать ошибки парсера, нужно заменить символ "<" ссылкой на сущность вот так:

```
<message>if salary &lt; 1000 then</message>
```

В XML есть пять изначально заданных сущностей:

ссылка на сущность	символ
<	< - меньше, чем
>	> - больше, чем
&	& - амперсанд
'	` - апостроф

`&quot;`	" - двойная кавычка

Ссылки на сущности всегда начинаются с амперсанда "&" и заканчиваются точкой с запятой: ";".

Обратите внимание: только символы "<" и "&" строго неприемлемы в XML. Апострофы, кавычки и символы "больше, чем" приемлемы, но лучше их все же заменять ссылками на сущности.

CDATA

Все, что находится внутри раздела CDATA игнорируется парсером

Если ваш текст содержит множество символов "<" или "&", например, программный код, можно задать секцию CDATA.

Секция CDATA начинается так: <![CDATA[" и заканчивается так: ends "]]>":

```
<script>
<![CDATA[
function matchwo(a,b)
{
if (a < b && a < 0) then
   {
   return 1
   }
else
   {
   return 0
   }
}
]]>
</script>
```

В этом примере все, что расположено внутри секции CDATA, будет игнорироваться парсером (не будет анализироваться).

Кодировка в XML

XML-документ может содержать символы различных языков мира
Чтобы парсер понимал эти символы, нужно сохранять документ в кодировке Unicode

Блокнот Windows 95/98 Notepad

Блокнот Notepad, встроенный в операционную систему Windows 95/98 не может сохранять документы в кодировке Unicode

Вы, конечно, можете использовать Notepad для того, чтобы редактировать и сохранять XML-документ, содержащий не-английские символы (например, русские):

```
<?xml version="1.0"?>
<note>
  <from>Jani</from>
  <to>Tove</to>
```

```
  <message>Russian: Привет!</message>
</note>
```

Но если вы сохраните этот документ, а затем попробуете его открыть в Internet Explorer, вы получите сообщение об ошибке.

Убедитесь сами:[58] (open xml)

Блокнот Windows 95/98 Notepad с кодировкой

При использовании Notepad, файлы должны сохраняться с указанным атрибутом encoding

Чтобы избегать возникновения ошибки, следует добавлять атрибут encoding в XML-декларацию вашего документа, но использовать при этом Unicode нельзя.

Эта кодировка НЕ вызовет сообщения об ошибке:[59] (open xml)

```
<?xml version="1.0" encoding="windows-1251"?>
```

Эта кодировка НЕ вызовет сообщения об ошибке:[60] (open xml)

```
<?xml version="1.0" encoding="ISO-8859-5"?>
```

Эта кодировка вызовет сообщения об ошибке:[61] (open xml)

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

Эта кодировка вызовет сообщения об ошибке:[62] (open xml)

```
<?xml version="1.0" encoding="UTF-16"?>
```

Блокнот Windows 2000 Notepad

Блокнот Windows 2000 Notepad может сохранять файлы в кодировке Unicode

Редактор Notepad, встроенный в операционную систему Windows 2000, поддерживает кодировку Unicode. Если вы сохраните этот XML-документ как Unicode (обратите внимание, что в документе не используется атрибут encoding),

```
<?xml version="1.0"?>
<note>
  <from>Jani</from>
  <to>Tove</to>
  <message>Russian: Привет!</message>
</note>
```

а затем попробуете его открыть в Internet Explorer, вы НЕ получите сообщение об ошибке.

Убедитесь сами:[63] (open xml)

Блокнот Windows 2000 Notepad с кодировкой

Файлы Windows 2000 Notepad, сохраненные как Unicode, используют кодировку "UTF-

58: http://xml.nsu.ru/xml/note_encode_none.xml
59: http://xml.nsu.ru/xml/note_encode_1251.xml
60: http://xml.nsu.ru/xml/note_encode_8859.xml
61: http://xml.nsu.ru/xml/note_encode_utf8.xml
62: http://xml.nsu.ru/xml/note_encode_utf16.xml
63: http://xml.nsu.ru/xml/note_encode_none_u.xml

16"

Если вы добавите атрибут encoding к файлу, сохраненному как Unicode, дополнительное указание иного значения кодировки вызовет ошибку.

Эта кодировка вызовет сообщения об ошибке:[64] (open xml)

```
<?xml version="1.0" encoding="windows-1251"?>
```

Эта кодировка вызовет сообщения об ошибке:[65] (open xml)

```
<?xml version="1.0" encoding="ISO-8859-5"?>
```

Эта кодировка вызовет сообщения об ошибке:[66] (open xml)

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

Эта кодировка НЕ вызовет сообщения об ошибке:[67] (open xml)

```
<?xml version="1.0" encoding="UTF-16"?>
```

Сообщения об ошибках

При загрузке XML-документа в Internet Explorer 5, вы можете получить два различных сообщения об ошибке, связанных с двумя различными проблемами:

В тексте документа был найден неподходящий символ.

Это сообщение появится, если в XML-документе имеется символ, не поддерживаемый кодировкой, указанной в атрибуте encoding. Обычно этот сообщение появляется, когда в XML-документе попадаются "иностранные" символы, а сам документ был сохранен в каком-либо одно-байтном редакторе, вроде Notepad, при этом атрибут encoding не был указан.

Переключение с текущей кодировки на указанную не поддерживается.

Это сообщение появится, если ваш файл был сохранен как Unicode/UTF-16, но в атрибуте encoding была задана какая-то одно-байтная кодировка, например Windows-1252, ISO-8859-1 или UTF-8. Также это сообщение появляется, когда документ был сохранен в одно-байтной кодировке, а в атрибуте encoding была указана дву-байтная кодировка, например, UTF-16.

Выводы

Вывод: атрибут encoding должен указывать кодировку, которая была использована, когда документ сохранялся. Самое лучшее, что можно делать, чтобы избегать ошибок: всегда сохраняйте XML-файлы как Unicode и не указывайте кодировку в атрибуте encoding.

Используйте редактор, который поддерживает Unicode (например, Windows 2000 Notepad) и не используйте атрибут encoding.

64: http://xml.nsu.ru/xml/note_encode_1251_u.xml
65: http://xml.nsu.ru/xml/note_encode_8859_u.xml
66: http://xml.nsu.ru/xml/note_encode_utf8_u.xml
67: http://xml.nsu.ru/xml/note_encode_utf16_u.xml

Простой XML-сервер

XML может генерироваться на сервере безо всяких установленных на нем средств управления XML

Хранение XML на сервере

XML-файлы могут храниться на вашем интернет-сервере

XML-файлы могут храниться на вашем интернет-сервере также, как и HTML-файлы.

Откройте редактор Notepad и напишите следующие строки:

```
<?xml version="1.0"?>
<note>
  <from>Jani</from>
  <to>Tove</to>
  <message>Remember me this weekend</message>
</note>
```

Все, что нужно сделать, чтобы этот XML-документ был готов к использованию - это сохранить его на интернет-сервере с каким-либо подходящим названием, например, "note.xml".

Обратите внимание: XML-файл должен находиться в той же директории, что и ваши HTML-файлы, а тип MIME XML-файлов должен быть установлен на "text/xml".

Генерирование XML с помощью ASP

XML можно генерировать на сервере без установки специального программного обеспечения для XML

Чтобы в ответ на запросы сервер выдавал XML, просто напишите следующий код и сохраните его на своем сервере в виде ASP-файла:

```
<%
Response.ContentType="text/xml"

Response.Write("<?xml version='1.0' ?>")
Response.Write("<note>")
Response.Write("<from>Jani</from>")
Response.Write("<to>Tove</to>")
Response.Write("<message>Remember me this weekend
</message>")
Response.Write("</note>")
%>
```

Обратите внимание, что параметр content type of the response должен быть установлен на XML. Посмотрите сами, как с сервера будет возвращаться ASP-файл:[68] (open xml)

(ASP - это файлы серверных скриптов, Active Server Pages. Если вы не знаете, как создавать ASP-файлы, вы можете посетить нашу школу ASP School):[69] (open link)

Получение XML из базы данных

68: http://www.w3schools.com/xml/note.asp
69: http://www.w3schools.com/asp/

XML можно генерировать из базы данных без установки специального программного обеспечения для XML

Предыдущий пример получения XML-документа легко можно преобразовать так, чтобы XML-документ получал данные из базы данных.

Чтобы сервер выдавал информацию из базы данных в ответ на запросы в виде XML, просто напишите следующий код и сохраните его в виде ASP-файла:

```
<%
Response.ContentType = "text/xml"

set conn=Server.CreateObject("ADODB.Connection")
conn.provider="Microsoft.Jet.OLEDB.4.0;"
conn.open server.mappath("../ado/database.mdb")
sql="select fname, lname from tblGuestBook"
set rs = Conn.Execute(sql)
rs.MoveFirst()

response.write("<?xml version='1.0' ?>")
response.write("<guestbook>")
while (not rs.EOF)
response.write("<guest>")
response.write("<fname>" & rs("fname") & "</fname>")
response.write("<lname>" & rs("lname") & "</lname>")
response.write("</guest>")
rs.MoveNext()
wend
rs.close()
conn.close()

response.write("</guestbook>")
%>
```

Вы можете посмотреть реальный ответ на запрос в базу данных, выдаваемый в виде XML:[70] (open xml)

(В этом примере используется ASP совместно с ADO. Если вы не знаете, как используется ADO, посетите нашу школу ADO School.): [71] (open link)

XML-приложения

В этом разделе приводится небольшой пример создания рабочей схемы для XML-приложения

Начинаем с XML-документа

Начинаем с простого XML-документа

Взгляните на наш оригинальный документ, каталог CD:

```
<?xml version="1.0"?>
```

70: http://www.w3schools.com/xml/guestbook.asp
71: http://www.w3schools.com/ado/

```
<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>
.
.
... more ...
.
```

С помощью Internet Explorer 5.0 и выше вы можете посмотреть полный документ:[72] (open xml)

Загрузите документ в островок данных (Data Island)

Островки данных используются для получения доступа к XML-файлу

Чтобы включить ваш XML-документ "внутрь" HTML-страницы, разместите на ней островок данных.

```
<xml src="cd_catalog.xml" id="xmldso" async="false">
</xml>
```

В этом примере XML-файл "cd_catalog.xml" будет загружен в "невидимый" островок данных, имеющий имя "xmldso". Атрибут async="false" добавляется в островок данных для того, чтобы обеспечить загрузку XML-данных еще до того, как начнется какая-либо обработка HTML-кода.

Привязка островка данных к HTML-таблице

Для отображения XML-данных можно использовать HTML-таблицу

Чтобы XML-данные отображались на вашей HTML-странице, нужно привязать ваш островок данных XML к HTML-элементу.

Чтобы привязать ваши XML-данные к HTML-таблице, добавьте к элементу table атрибут datasrc (источник данных) и добавьте атрибуты datafld (поле данных) к элементам внутри таблицы:

```
<table datasrc="#xmldso" width="100%" border="1">

<thead>
<th>Title</th>
<th>Artist</th>
<th>Year</th>
</thead>

<tr align="left">
<td><span datafld="TITLE"></span></td>
<td><span datafld="ARTIST"></span></td>
<td><span datafld="YEAR"></span></td>
</tr>
```

72: http://xml.nsu.ru/xml/cd_catalog.xml

```
</table>
```

Вы можете посмотреть сами, как XML-данные отображаются в таблице:[73] (open editor)

Привязка островка данных к элементам или

Для отображения XML-данных можно использовать элементы и

Для отображения XML-данных не обязательно использовать таблицу. Данные из островка данных могут отображаться в любом месте HTML-страницы.

Все, что нужно сделать - это добавить на страницу элементы или . При этом используйте атрибут datasrc для того, чтобы связать элементы с островком данных и атрибут datafld для того, чтобы связать каждый конкретный HTML-элемент с XML-элементом, например, так:

```
<br />Title:
<span datasrc="#xmldso" datafld="TITLE"></span>
<br />Artist:
<span datasrc="#xmldso" datafld="ARTIST"></span>
<br />Year:
<span datasrc="#xmldso" datafld="YEAR"></span>
```

Или так:

```
<br />Title:
<div datasrc="#xmldso" datafld="TITLE"></div>
<br />Artist:
<div datasrc="#xmldso" datafld="ARTIST"></div>
<br />Year:
<div datasrc="#xmldso" datafld="YEAR"></div>
```

Вы можете посмотреть сами, как XML-данные отображаются в HTML-элементах:[74] (open editor)

Обратите внимание, что если вы используете элемент , данные отображаются с новой строки.

В приведенных выше примерах отображается только одна запись XML-данных. Чтобы переходить к следующей записи, в код нужно добавить специальный скрипт.

Добавление в XML скрипта навигации

Навигация выполняется скриптом

Чтобы добавить возможность навигации по островку XML-данных, создайте скрипт, который вызывает методы островка данных movenext() и moveprevious():

```
<script type="text/javascript">
function movenext()
{
x=xmldso.recordset
if (x.absoluteposition < x.recordcount)
        {
        x.movenext()
        }
```

73: http://www.w3schools.com/xml/tryit.asp?filename=cd_list
74: http://www.w3schools.com/xml/tryit.asp?filename=cd_first

```
}
function moveprevious()
{
x=xmldso.recordset
if (x.absoluteposition > 1)
        {
        x.moveprevious()
        }
}
</script>
```

Вы можете посмотреть сами, как действует навигация по островку XML-данных:[75] (open editor)

Теперь все вместе

С небольшой фантазией вы можете разработать полноценное XML-приложение

Если вы примените все, что вы узнали на этой школе и добавите немного фантазии, вы легко разовьете этот пример в полноценное приложение.

Посмотрите сами, как можно добавить фантазию к этому примеру:[76] (open editor)

XML HTTP-запросы

XML-данные могут быть запрошены у сервера с применением HTTP-запроса

Запрос браузера

Браузер с помощью HTTP-запроса может запросить у сервера XML:

```
var objHTTP = new ActiveXObject("Microsoft.XMLHTTP")
objHTTP.Open('GET','httprequest.asp',false)
objHTTP.Send()
```

Чтобы увидеть результат запроса, вы можете его отобразить в своем браузере:

```
document.all['A1'].innerText= objHTTP.status
document.all['A2'].innerText= objHTTP.statusText
document.all['A3'].innerText= objHTTP.responseText
```

Попробуйте сделать это сами с применением JavaScript:[77] (open editor)

Попробуйте сделать это сами с применением VBScript:[78] (open editor)

Коммуникация с сервером

С помощью HTTP-запросов вы можете вступать с сервером в коммуникации

Коммуникация с сервером с использованием XML:[79] (open editor)

75: http://www.w3schools.com/xml/tryit.asp?filename=cd_navigate
76: http://www.w3schools.com/xml/tryit.asp?filename=cd_application
77: http://www.w3schools.com/xml/tryit.asp?filename=httprequest_js
78: http://www.w3schools.com/xml/tryit.asp?filename=httprequest_vb
79: http://www.w3schools.com/xml/tryit.asp?filename=tryxml_send

В этом примере результат запроса "подделывается" на сервере с помощью ASP-скрипта:

```
<%
response.ContentType="text/xml"
txt="<answer><text>12 Years</text></answer>"
response.write(txt)
%>
```

То есть, результатом запроса будет всегда "12 years", не важно, какой запрос был сделан. В реальных условиях вам понадобится написать специальный код, который будет анализировать запрос и давать на него правильный ответ.

Поведения для HTML и XML - новый DHTML?

Поведение (behavior) - это атрибутный CSS-селектор. Он указывает на XML-файл, содержащий код, который должен быть выполнен в связи с какими-то элементами на веб-странице
Технология Behaviors не является стандартом W3C, она применяется только Microsoft

Поведения - что это такое?

Поведение - это новый атрибутный CSS-селектор

Селектор поведения может указывать на отдельный XML-файл, содержащий код, который должен быть выполнен в связи с определенными XML или HTML-элементами на веб-странице.

Вы понимаете? Метод, позволяющий совершенно убрать с HTML-страниц скриптовый код. Прекрасно! Теперь можно начать писать библиотеки скриптов и привязывать скрипты к любому элементу, которому мы захотим!

Как они работают?

Взгляните на этот HTML-файл. В нем имеется элемент <style>, который задает поведение для элемента <h1>:

```
<html>
<head>
<style>
h1 { behavior: url(behave.htc) }
</style>
</head>

<body>
<h1>Move your Mouse over me</h1>
</body>
</html>
```

Попробуйте этот пример в действии, поводите мышью над текстом:[80] (open editor)

Код поведения хранится в XML-документе behave.htc:

```
<component>
<attach for="element" event="onmouseover"
 handler="hig_lite" />
<attach for="element" event="onmouseout"
```

80: http://www.w3schools.com/xml/tryit.asp?filename=behave

```
 handler="low_lite" />

<script type="text/javascript">
   function hig_lite()
   {
   element.style.color=255
   }
   function low_lite()
   {
   element.style.color=0
   }
</script>
</component>
```

Файл поведения содержит JavaScript. Скрипт находится внутри элемента <component>. Последний также содержит описание событий, управляющих скриптом (handlers). Интересное поведение, не правда ли?

Связанные с XML технологии

В этой главе приводится список технологий, которые важны для понимания и разработки XML-приложений
На этот список также можно смотреть как на примерный учебный план, если вы решили заняться XML всерьез

Технологии XML

XHTML - расширяемый (extensible) HTML[81] (open link)

XHTML - это переформулировка языка HTML 4.01 по стандарту XML. Язык XHTML 1.0 - это новейшая версия языка HTML.

CSS - каскадные таблицы стилей (cascading style sheets)[82] (open link)

К XML-документам можно добавлять таблицы CSS, чтобы задать отображение отдельных элементов

XSL - расширяемый язык таблиц стилей (extensible style sheet language)

XSL состоит из трех частей: языка преобразования XML-документов (XML Document Transformation) (именуемый XSLT, см. ниже), правил синтаксиса соответствия паттернов (положений в дереве документа) (именуемый XPath, см. ниже), и интерпретатора форматирования объектов.

XSLT - язык XML-преобразований (XML Transformation)[83] (open link)

XSLT - гораздо более мощное средство, чем CSS. Его можно использовать для преобразования XML-документа во множество различных форматов.

XPath - язык соответствия паттернов (XML pattern matching)

XPath - это язык адресации к различным частям XML-документа. Он был создан для использования вместе с языками XSLT и XPointer.

XLink - язык XML-ссылок (XML linking language)

XLink позволяет вставлять внутрь XML-документа элементы, создающие связи между XML-ресурсами.

81: http://www.w3schools.com/xhtml/
82: http://www.w3schools.com/css/
83: http://xml.nsu.ru/xsl/xsl_home.xml

XPointer - язык XML-указателей (XML pointer language)

XPointer помогает устроить адресацию ко внутренним структурам XML-документов, таким как элементам, атрибутам и содержанию элементов.

DTD - определение типа документа (document type definition)[84] (open link)

DTD используется для определения допустимых строительных блоков XML-документа.

Пространства имен

Пространства имен XML предоставляют метод определения имен элементов и атрибутов, используемых в XML, привязкой к ним определенных URI-ссылок.

XSD - язык XML-схем[85] (open link)

Схемы - мощная альтернатива DTD. Схемы пишутся на XML, поддерживают пространства имен и типизацию данных.

XDR - XML Data Reduced

XDR - это усеченная версия XML-схем. Internet Explorer 5.0 был создан с поддержкой XDR в то время, когда стандарт XML-схем еще находился в стадии разработки. Microsoft пообещала реализовать полную поддержку XML-схем, когда эта спецификация получит статус рекомендации консорциума W3C.

DOM - объектная модель документа (document object model)[86] (open link)

DOM определяет средства взаимодействия, свойства и методы манипулирования с XML-документами.

XQL - язык XML-запросов (XML query language)

XQL поддерживает средства создания запросов для извлечения данных из XML-документов.

SAX - Simple API for XML

SAX - это еще один интерфейс для чтения и манипуляций с XML-документами.

Рекомендации W3C

Консорциум W3C (World Wide Web Consortium) был создан в 1994 году, чтобы руководить разработкой общих WWW-протоколов, таких, как HTML, CSS и XML.

Наиболее важная деятельность консорциума - это разработка веб-спецификаций (они называются рекомендациями), которые описывают протоколы (такие, как HTML и XML) и другие составные части WWW.

Вы можете узнать больше о статусе каждого стандарта XML на наших школах W3C School:[87] (open link)

Примеры XML

Просмотр XML-файлов

Простой XML-файл (note.xml):[88] (open xml)

84: http://xml.nsu.ru/dtd/dtd_home.xml
85: http://xml.nsu.ru/schema/schema_home.xml
86: http://xml.nsu.ru/dom/dom_home.xml
87: http://www.w3schools.com/w3c/
88: http://xml.nsu.ru/xml/note.xml

Тот же самый XML-файл, содержащий ошибку:[89] (open xml)

Каталог компакт-дисков, написанный на XML:[90] (open xml)

XML-каталог растений:[91] (open xml)

Меню на XML:[92] (open xml)

Просмотр XML-файлов, использующих DTD

Файл note.xml, имеющий внутренний DTD:[93] (open xml)

Файл note.xml с внешним DTD:[94] (open xml)

XML-парсер Microsoft

Простой XML-файл (xml_note.xml):[95] (open xml)

Загрузка этого файла в парсер:[96] (open editor)

Передача парсеру дерева узлов XML-документа:[97] (open editor)

Загрузка того же файла внутрь HTML:[98] (open editor)

Показ XML с помощью JavaScript

Простой XML-файл (note.xml):[99] (open xml)

Форматирование этого файла с применением JavaScript:[100] (open editor)

XML и CSS

Каталог компакт-дисков, написанный на XML:[101] (open xml)

Предназначенный для этого документа CSS-файл:[102] (open css)

XML-каталог компакт-дисков, отформатированный с помощью CSS-файла:[103] (open xml)

XML и XSL

Меню на XML:[104] (open xml)

Предназначенный для этого документа XSL-файл:[105] (open xsl)

Меню, отображаемое при участии таблицы стилей XSL:[106] (open xml)

Привязка данных

Каталог компакт-дисков, написанный на XML:[107] (open xml)

89: http://xml.nsu.ru/xml/note_error.xml
90: http://xml.nsu.ru/xml/cd_catalog.xml
91: http://xml.nsu.ru/xml/plant_catalog.xml
92: http://xml.nsu.ru/xml/simple.xml
93: http://xml.nsu.ru/xml/node_in_dtd.xml
94: http://xml.nsu.ru/xml/node_ex_dtd.xml
95: http://xml.nsu.ru/xml/xml_note.xml
96: http://www.w3schools.com/xml/tryit.asp?filename=node_parsertest
97: http://www.w3schools.com/xml/tryit.asp?filename=xml_note_traverse
98: http://www.w3schools.com/xml/tryit.asp?filename=xml_note
99: http://xml.nsu.ru/xml/xml_note.xml
100: http://www.w3schools.com/xml/tryit.asp?filename=xml_note
101: http://xml.nsu.ru/xml/cd_catalog.xml
102: http://xml.nsu.ru/xml/cd_catalog.txt
103: http://xml.nsu.ru/xml/cd_catalog_with_css.xml
104: http://xml.nsu.ru/xml/simple.xml
105: http://xml.nsu.ru/xml/simple.xsl
106: http://xml.nsu.ru/xml/simplexsl.xml
107: http://xml.nsu.ru/xml/cd_catalog.xml

Привязка каталога CD к HTML-таблице:[108] (open editor)

Добавление элементов <thead>, <tfoot>, <tbody>:[109] (open editor)

Результат запроса в базу данных

Результат запроса в базу данных на XML:[110] (open editor)

Отображение в виде HTML

Каталог компакт-дисков, написанный на XML:[111] (open xml)

Каталог можно отображать внутри HTML-элементов:[112] (open editor)

Каталог отображается в HTML-таблице:[113] (open editor)

По CD-каталогу можно устроить навигацию:[114] (open editor)

Пример простого приложения, работающего с каталогом:[115] (open editor)

Запрос XML-данных у сервера

Запрос XML у сервера с применением JavaScript:[116] (open editor)

Запрос XML у сервера с применением VBScript:[117] (open editor)

Отправка запроса на сервер:[118] (open editor)

Коммуникация с сервером с применением XML:[119] (open editor)

Поведения

Поведения:[120] (open editor)

Developed by Metaphor (c) 2002

108: http://www.w3schools.com/xml/tryit.asp?filename=cd_catalog_island
109: http://www.w3schools.com/xml/tryit.asp?filename=cd_catalog_island_thead
110: http://www.w3schools.com/xml/guestbook.asp
111: http://xml.nsu.ru/xml/cd_catalog.xml
112: http://www.w3schools.com/xml/tryit.asp?filename=cd_first
113: http://www.w3schools.com/xml/tryit.asp?filename=cd_list
114: http://www.w3schools.com/xml/tryit.asp?filename=cd_navigate
115: http://www.w3schools.com/xml/tryit.asp?filename=cd_application
116: http://www.w3schools.com/xml/tryit.asp?filename=httprequest_js
117: http://www.w3schools.com/xml/tryit.asp?filename=httprequest_vb
118: http://www.w3schools.com/xml/tryit.asp?filename=xml_send
119: http://www.w3schools.com/xml/tryit.asp?filename=tryxml_send
120: http://www.w3schools.com/xml/tryit.asp?filename=behave