Лекция 6

Технология XML

Extensible Markup Language (XML) е бързо развиваща се технология за мощни приложения използвани за представяне, управление и редакция на данни. Съвместно с езика за представяне (XSL) и стандартизирания документен обектен модел (Document Object Model – DOM), XML представлява технология за форматизиране както във Web, така и за обикновени приложения.

XML е технология, която се занимава с описание и структуриране на данните. Данните се съхраняват в компютърната памет или като двоични или като текстови файлове. Двоичните файлове представляват поредица от двоични цифри, които се интерпретират от програмното приложение. Поради тази причина, двоичните файлове могат да се създават и четат само от определени програми. Например, когато текстообработваща програма създаде документ, тя създава двоичен файл със собствен формат. В него по специален начин се се отбелязва удебеляването на част от текста, преминаване към нов параграф или страница и редица други обозначения. Тези специализирани обозначения се могат да се интерпретират само от текстообработващата програма. Вмъкнатите в документа кодове за обозначаване на различни характеристики на текста се наричат метаданни (metadata). В тези метаданни се крие разликата между различните типове файлове.

Недостатък на двоичните файлове е че те са "частни" и се отнасят само за определено програмно приложение. Повечето програми за текстообработка имат транслатори, които могат да преобразуват форматите на други текстообработващи програми. Предимството на двоичните файлови формати е, че компютрите лесно и бързо обработват файлове с метаданни.

В текстовите файлове, двоичните данни са групирани по определен стандартен начин, така че винаги да представят числа явяващи се кодове на определени символи. Поради въведените стандарти за кодиране на текстова информация, текстовите файлове могат да се четат от много приложения. Ако се създаде текстов документ, той може да се прочете с помоща на произволен текстов редактор. По този начин информацията се поделя много лесно между различни потребители.

Както се вижда двоичните и текстови файлове имат предимства и недостатъци. Би било добре, ако съществува един формат, който да комбинира преимуществата на двата типа формати. Идеята за универсален формат за данни не е нова. Тя съществува още от самото начало на развитие на компютърните системи. Един от първите опити да се комбинира универсален формат с богати възможности за съхранение е SGML (Standard Generalized Markup Language). Това е текстово базиран език, който може да се използва за форматиране (mark-up) на данни по един сомоописателен начин.

Най-известното приложение на SGML е HTML. Тъй като правилата за създаване на SGML документи са доказали своята ефективност е напълно нормално да се създаде един специфичен речник (HTML), който да се използва като универсален език за форматиране. Идеята е била всеки HTML документ (Web страница) да може да се представи във всяко приложение, което разбира HTML (браузър). Браузерите е трябва да може не само да показва документа, но ако документът съдържа хипервръзка към други документи да покаже и тях.

За нещастие, SGML е доста сложен език. Въпреки големият успех на HTML, той има сериозни ограничения. Той може само да показва документи, но не съдържа подробности за описваните в документа елементи и обекти. Това затруднява в много голяма степен, търсенето на информация в Internet. Машините за търсене в Internet, не могат да прочитат всички документи намиращи се в мрежата за да открият някаква информация. Те

преглеждат само заглавната част на HTML документите, но в много случаи, тя е недостатъчна.
Поради тази причина е разработен Extensible Markup Language (XML). XML е подмножество на SGML за форматиране на данни, от който са премахнати някои от по-сложните елементи. Той е проектиран да бъде напълно съвместим с SGML, което означава, че всеки документ създаден с XML синтаксис ще може да се прочете и с SGML инструменти. Обратната съвместимост не е гарантирана, все пак XML е подмножество на SGML.
Важно е да се отбележи, че XML не е език за форматиране, а е стандарт за създаване на езици, които отговарят на XML критерии. Казано по-просто, XML описва синтаксиса, който да се използва при създаване на конкретни езици за форматиране. Например, да предположим, че съществуват данни за някакво име (човек) и трябва тези данни да се споделят с други хора, като се публикуват в Internet. Освен това, в структурата от данни трябва да се отдели името и то да се използва като указател за търсене на информация за това име. В HTML формат документа би изглеждал така:

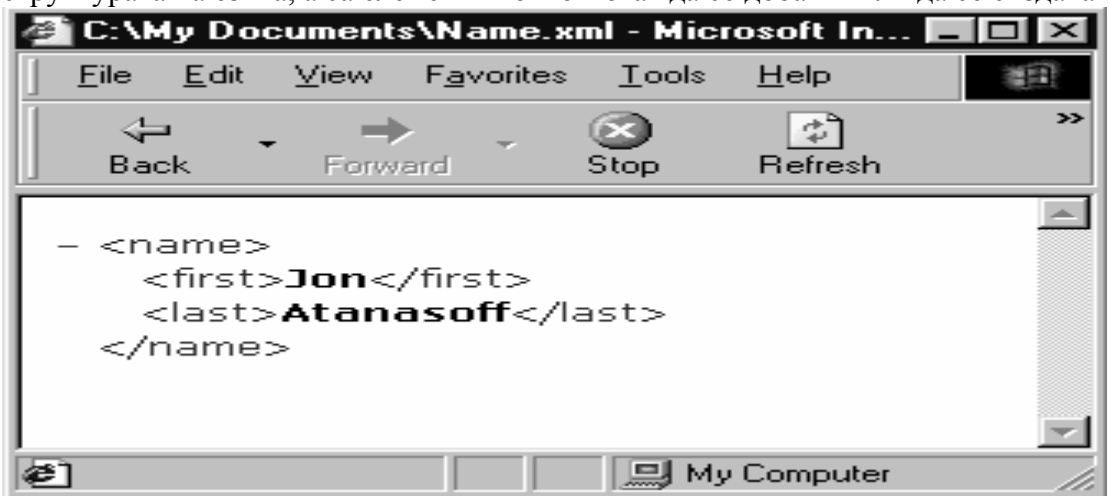
```
<HTML>
<HEAD><TITLE>Name</TITLE></HEAD>
<BODY>
<P> Jon Atanasoff</P>
<BODY>
<HTML>
```

Тук информацията за името е записана като параграф в HTML документа и една машина за търсене, не може да разбере, че данните в този параграф са име на човек.
В един XML документ информацията за името може да се запише във вида:

```
<name>
     <first> Jon</first>
     <last> Atanasoff</last>
</name>
```

От този прост пример може да се разбере, защо езиците за форматиране от рода на HTML и XML се наричат самоописателни. Гледайки данните, може лесно да се каже, че информацията се отнася за <**name**> (име). Може да се види, че има данни наречени <**first**> (първо) и други наречени <**last**> (последно). Тези данни могат да се именоват по произволен начин, но ако ще се използва такъв начин на структуриране на данните е добре да се използват имен които имат някакво значение.
XML версията на даден документ, е по-голяма отколкото с обикновен текст или с HTML. Използването на XML за форматиране на данни увеличава размера на документите, но малкият размер на документите не е цел на XML. Той има за цел да улесни писането на софтуер, който осъществява достъп до данни, като описва структурата на данните. Предимствата за лесен начин на писане на програмен код са по-големи от недостатъците от увеличаване на размера на документите.
XML документите могат да бъдат разглеждани от съвременните браузъри. Например, посредством Internet Explorer горният XML текст, записан като '.xml' файл може да се изобрази и би изглеждал както е показано на фиг.6.4.
Въпреки че във файла няма информация за начина по-който да бъдат показани данните, Internet Explorer го форматира, като показва данните с удебелен шрифт, а таговете с друг цвят. Освен това, <**name**> може да се свива и разширява (знака '-'), както това се прави с папките в Windows Explorer. Това е много удобно за големи XML документи, когато трябва да се съсредоточи вниманието върху някое подмножество от данни.
За да се представи само истинската информация, без форматиращите данни (метаданни) е необходима специална таблица със стилове, в която са описани стиловете за всички тагове

използвани в документа. Но XML не е само за представяне на данни, а и за много други цели. Поради тази причина, стиловете, въпреки че са много важна част, не са вградени в структурата на езика, а са елементи които могат да се добавят или да се създават.



Фиг. 6.4 Internet Explorer вид на XML document

XML е гъвкава система и може да се използва като основа за дефиниране на езици за обмен на данни, особено при комуникация чрез Internet. Освен че улеснява работата с данни в приложения, които трябва да извършат достъп до данни (данните <**name**> в горния пример), XML улеснява и споделянето на информацията с други потребители. Трансфериран по Internet, XML файлът може да се използва дори без обработка, само като текст. Форматирането на данните може да се извършва свободно, с произволно създадени правила за структуриране на данните. Разбира се предимствата на XML се изявяват най-добре, когато се използват едни и същи формати за общи дейности. Вече съществуват множество проекти за създаване на стандартни за индустрията **речници (vocabularies**) за описване на разнообразни типове данни.

1. Йерархия в XML

XML групира информацията също както и HTML в йерархични структури. Взаимовръзката между елементиете в един документ са от типа **родител/наследник** и **брат/сестра**. Ако разгледаме представеният пример с типа данни <**name**>, можем да установим, че <**name**> е родителски елемент по отношение на елементите <**first**> и <**last**>. Елементите <**first**> и <**last**> могат да се разглеждат, като наследници (дъщерни елементи) на <**name**>, а между <**first**> и <**last**> съществува връзка от тпа брат/сестра. Трябва да отбележи, че данните се явяват наследници на съответните елементи. Например Jon е наследник на елемента <**first**>, а Atanasoff на елемента <**last**>.

Така разгледана структурата на данните в XML документ се нарича дърво. Всяка част от дървото, която има наследници се нарича клон, докато частите, които нямат наследници се наричат листа.

Голямата популярност на XML се дължи на способността му да описва произволна информация. Той е много гъвкав по отношение на структурирането на данните. За описване на структурата на данните се използва определен синтаксис.

2. XML парсери

Парсерите са програми, които могат да четат XML синтаксиса и да извличат информация от XML документа. Потребителят може да използва парсери в собствените си програми, така че XML данните не се използват директно, а обработени от съответните парсери. По

този начин, тежката и сложна част от обработката и подреждането на данните се извършва от специализираните програми – парсери.
Съществуват парсери и за SGML документи, но те са много сложни. Тъй като XML е подмножество на SGML, много по-лесно е да се напише парсер за XML, отколкото за SGML.
Парсерите са програми, които не се занимават с конкретнот съдържание на форматиращите елементи (таговете). Те анализират синтактическите правила и структурата на данните в документа и подреждат информацията в нейната йерархическа последователност. Това означава, че парсерите са универсални програми, които могат да обработват произволни XML документи, независимо с каква цел са създадени документите и по какъв начин те трябва да се показват.
Процесът на обработка на XML документи с помоща на парсери се нарича парсване на XML. Приема се, че XML процесорът (парсер) изпълнява обработката на XML документите в полза на друг модул или приложение. Съществуват доста голям брой парсери, като много от тях са безплатни: Microsoft Internet Explorer Parser, DataChannel XJ Parser, IBM xml4j и др.
В XML спецификацията е указано не само как процесорът (парсерът) трябва да извлича необходимата информация от даден XML документ, но също така и как трябва да обработва грешки в XML кода. В XML спецификацията са посочени два типа грешки: **грешки** и **фатални грешки**.
Грешките представляват нарушаване на правилата на спецификацията, което води до неопределени резултати. Обикновено на XML парсерите е разрешено да продължат парсването след грешката, но те извеждат съобщение за дадената грешка.
Фаталните грешки са по-сериозни и според спецификацията, парсерът не може да продължи обработката на документа. Всяка грешка, която нарушава конструкцията на XML документа е фатална грешка. Причината за този драстичен подход при среща на фатална грешка е че разработчиците на парсери трудно биха създали код, който да анализира неправилно конструиран документ

3. HTML и XML

Това което HTML прави за показване на данните във Web страницата е това, което XML прави за обмен на данните. Фундаменталната разлика между HTML и XML се изразява в следното:

- HTML е предназначен за специфични приложения – да се доставя информация, най-често визуално, посредством браузъри;
- XML няма специфично приложение, той е предназначен за извършване на доста по-широк кръг от дейности свързани с форматирането на данни.

Това е много важна концепция. Тъй като HTML има специфично приложение, той има и краен набор от специални конструкции за форматиране на текст. На теория, всеки Web браузър може да разпознае един HTML документ, защото в крайна сметка той трябва да провери правилността на краен брой тагове.
От друга страна, ако се създаде един XML документ, то всеки XML парсер може да извлече информация документа, но не може да се гарантира, че приложението ще разбере тази информация. Това е така, защото парсерът може да проследи структурата на данните, но не е сигурно че програмните приложения могат да се възползват от тази структура (или да разберат тази структура).
Следователно, може да се създават XML документи за описание на всякаква информация, но преди да се каже че XML документите са полезни, трябва да са създадени приложения, които да могат да се възползват от тази информация.

4. Синтактични правила в XML

След като бяха представни някои основни характеристики на технологията XML, е нобходимо да се представят и синтактичните правила, по които се съставя един XML документ. Нека разгледаме един фрагмент от XML документ:

```
<name>
     <first> Jon</first>
     <middle> Vinsent </middle>
     <last> Atanasoff</last>
</name>
```

Думите намиращи се между ъгловите скоби < и > се наричат XML тагове. Данните (информацията) в един XML документ се разполага между разнообразни тагове, които съставляват форматиращата част на документа.

Както може да се установи от горния пример, таговете се използват по двойки, на всеки отварящ таг съотвтства затварящ таг. Затварящите тагове се различават от започващите с това, че след отварящият знак '<' следва знакът '/', който липсва при отварящите тагове. В този смисъл XML таговете се използват, както таговете в HTML, но има известни различия, които ще бъдат представени по-късно.

Цялата информация от началото на отварящият таг до завършващия таг се нарича елемент. Например **<first> Jon</first>** от горният пример е елемент.Текстът между отварящият и затварящият таг се нарича съдържание на елемента. Когато съдържанието на елемента представлява само данни то се нарича **Parsed Character DATA** (парсвани знакови данни), като по-често се използва съкращението **PCDATA**. Целият документ започващ с **<name>** и завършващ с **</name>,** също представлява елемент, който съдържа други елементи. Такъв елемент се нарича главен **(root**) елемент.

В XML документите трябва да се спазват определени синтактични правила. Те се различават от правилата използвани в HTML.

- **Всеки елемент трябва да има отварящ и затварящ таг**. В основната спецификация SGML не всеки елемент трябва да има отварящ и затварящ таг. Това се отнася и за HTML. HTML браузърите са достатъчно умни, за да разбират логическото място на някои отсъстващи тагове. Това е така, защото HTML съдържа краен брой тагове и може да се направи обработваща програма, която да проверява логиката на документ съставен от краен брой правила. При XML, създаването на парсер, който да отчита логика на произволно подбраната система от тагове е изключително сложна (почти невъзможна) задача.

-**Наименования на елементите**. Създаването на елементи в XML изисква те да се именоват. XML, за разлика от повечето програмни езици, няма никакви запазени думи, които трябва да се избягват при съставяне на имената на елементите. Все пак съществуват някои правила при задаване на имената на елементите. Имената на елементите трябва да започват с букви (включително и не латински) или със знака '_', но не с числа или други знаци. След първата буква може да се използват цифри, както и знаците '-','.'. Имената не могат да съдържат интервали. Имената не могат да започват с буквеното съчетание 'xml', независимо с главни или малки букви. Не може да има интервал след отварящият знак '<', но преди затварящият може да има.

- **Чувствителност към регистъра**. Важен елемент от синтаксиса на XML е чувствителността към регистъра. Това е голямата разлика с HTML, който не различава малките и големи букви в регистъра. Следователно, за един XML документ тагът **<first>** е различен от **<FIRST>**, който пък е различен от **<First>.** В това отношение XML прилича на някои от програмните езици: C, C++, Java, докато HTML прилича на регистрово незавизимите езици какъвто е Pascal.

- Атрибути. Освен тагове и елементи, XML документите могат да съдържат и атрибути. Те предсравляват двойки име/стойност, които се асоциират с даден елемент. Атрибутите се прикачват към отварящият таг, както е показано в следния фрагмент от XML документ:

```
<name nickname='Bobi'>
     <first>Robert</first>
     <last> Cenedy</last>
</name>
```

Атрибутите трябва да имат стойност – дори тази стойност да представлява празен текстов низ “”. За задаване на текстов низ могат да се използват и апострофи и кавички. Правилата за задаване на имена на атрибутите са същите както за елементите.

В XML обществото, съществува дебат за необходимостта от използване на атрибути. Привържениците на използване на атрибутите в структурата на XML документи, акцентират върху възможността за представяне на метаданни, които не са важни за голяма част от приложенията. По този начин, логически данните се разделят на групи по важност. В горният пример вместо атрибут може да се използва още един елемент равнозначен на `<first>` и `<last>`, но те ще са еднакви по важност (ранг).

- Коментари. Коментарите позволяват да се вмъква в XML документите текст, който в действителност не се явява част от документа, а е предназначен за пояснения върху структурата на документа. Коментарите започват с символния низ ‘**<!-**’ и завършва с ‘**->**’. Вмъкването на коментари изисква спазването на някои правила. Първо, не може да се поставя коментар в таг и второ не е разрешено използването на символа ‘-‘ в коментар. Коментарите се изключват от текста, който се обработва с XML парсерите.

- Празни елементи. Понякога един елемент може да не съдържа данни. Ето един пример, който вече използвахме:

```
<name>
     <first> Jon</first>
     <middle> </middle>
     <last> Atanasoff</last>
</name>
```

В случая елементът `<middle> </middle>` не съдържа данни. Той може да се запише в горният текст във вида `<middle/>`. Това е единственият случай, в който отварящият таг не се нуждае от затварящ таг, защото двата тага са слети в един. Във всички други случаи таговете трябва да са по двойки.

Празните елементи в действителност не дават нищо за документа. Едно от местата, където често се използват празни елементи е когато цялата информация се съдържа в атрибути. Ако искаме да пренапишем горният XML фрагмент, като данните запишен с атрибути той би изглеждал по следния начин:

```
<name first=”Jon” last=”Atanasoff” />
```

Друг, често срещан пример на използване на празен елемент, е когато само наименованието на тага е достатъчно за задаване на елемента – например HTML за нов ред **
** може да се преобразува в XML празен елемент **
**.

- XML декларация. Често е удобно да се идентифицира конкретен тип документ. XML предоставя XML декларация, посредством която може да се обозначи даден документ като XML и заедно с това се даде допълнителна информация за парсерите, които обработват документа. XML декларацията не е задължителна, но тя е полезна и е за предпочитане да се включи в документа. Една типична XML декларация изглежда по следния начин:

```
<?xml version='1.0' encoding='UTF-16' standalone='yes'?>
```

От представения пример могат да се установят някои правила за задаване на XML декларация. На първо място, XML декларацията започва със знаковия низ **'<?xml**' и завършва с '**?>**'. При използване на XML декларация трябва да се включи версия на XML (**version**). Другите атрибути **encoding** и **standalone** не са задължителни. Атрибутите **version, encoding** и **standalone** трябва да бъдат в този ред, както са показани в примера.
На този етап версията трябва да е 1.0. Ако се използва друг номер на версия, парсерите написани за версия 1.0 би трябвало да отхвърлят документа. XML декларацията трябва да се намира в началото на документа.
Атрибутът **encoding** се използва в XML декларацията за да информира парсера, който обработва документа, каква кодова таблица за представяне на текстова информация е използвана. XML парсерът прочита документа със зададената кодировка и я преобразува в Unicode. Ако не е зададена никаква кодировка, тя се приема че от вида UFT-16 (Unicode - използват се два байта за един символ) или UFT-8 (1 байт за един символ).
Ако атрибут **standalone** е включен в XML декларацията, той трябва да има стойност '**yes**' или '**no**'. Когато е указана стойност 'yes', документът съществува самостоятелно и не зависи от други файлове. Когато стойността е 'no' означава, че документът може да зависи от други файлове.

- **Инструкция за обработка**. Понякога се налага да се вгради в документа с данните, информация за специфични инструкции, които да дадат указание как да се обработи информацията. XML предоставя един механизъм, осигуряващ използването на инструкции за обработка, който се нарича **Processing instructions (Pis) – инструкции за обработка**. Чрез него може да се въведат инструкции, които не се явяват част от документа, а се подават на приложението, което използва документа. В действителност не съществуват много правила за PI инструкциите. В основат си те просто са последователност от текстовите символи "**<?**", след което се задава наименованието на приложението, което трябва да получи инструкцията (**PITarget**), произволна инструкция и накрая се поставя знак "**?**>". Ето един пример на инструкция за обработка: `<?nameprocessor SELECT * FROM blah?>`. Тук **PITarget** е **nameprocessor**, а текстът на инструкцията е **`SELECT * FROM blah.`**

- **Забранени PCDATA знакове**. Има някои запазени знаци, които не могат да се включат в данните за даден елемент (PCDATA), защото те се използват в синтаксиса на XML. Такива са например знаковете "**<**" и "**&**", които в секцията за данни са необходими при запис на някои операции за сравнение:

```
<!- Това е грешен XML текст ->
<comparision> 6 is < 7 & 7 > 6 </comparision>
```

Ако се отвори този XML текст например с Internet Explorer, ще се получи съобщение за грешка. Това е така, защото парсерът преминавайки през символа '<' в данните очаква име на някакъв таг, а не празно поле (или число). Същото се получава и при среща на знака '&'. Съществуват два начина да се заобиколи това ограничение – да се промени интерпретацията на символите или да се използва CDATA секция в документа.
За да се промени интерпретацията на символите "**<**" и "**&**" трябва всеки символ "<" да се замени с "<", ">" с > и "&" с "&". Тогава горният грешен израз може да се запише по следният начин:

```
<comparision> 6 is &lt; 7 &amp; 7 &gt; 6 </comparision>
```

Въведените алтернативни означения на забранените знаци се наричат референции към обект. Може да се променя интерпретацията и на други знаци с помоща на знакови референции.
Ако в даден документ се съдържат много от забранените символи, писането на документа става много трудно, а и той би изглеждал доста нечетлив. Затова, в такива случаи се използва секция CDATA. Това е един наследен от SGML термин – съкращение от

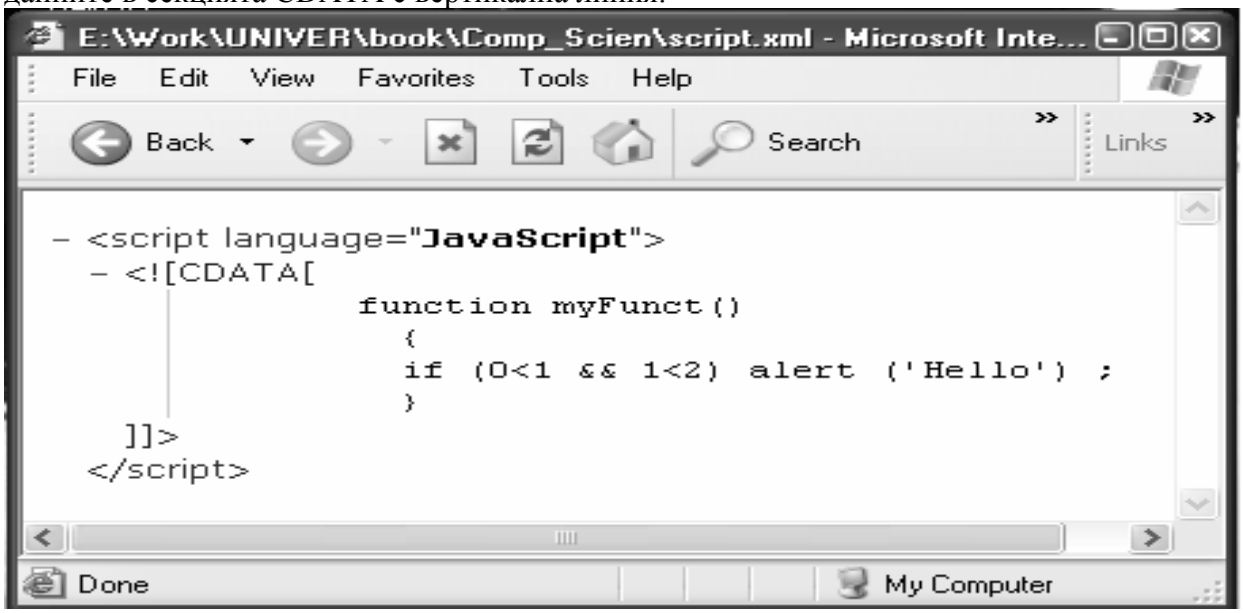
Character DATA. С помоща на CDATA секция може да се каже на XML парсера да не интерпретира (не парсва) текста, намиращ се в обхвата на CDATA секцията. CDATA секцията изглежда по следният начин:

```
<comparision> <![CDATA[6 is < 7 & 7 > 6]] </comparision>
```

Всичко, което започва след <!CDATA[и завършва с]]>, се игнорира от парсера и се подава в непроменен вид на приложението, което ще използва данните. CDATA секциите се използват, за да се изключат от парсване и дрги части от текста на XML документ, например програмен код написан на JavaScript:

```
<script language="JavaScript"><![CDATA[
     function myFunct()
       {
       if (0<1 && 1<2) alert ('Hello') ;
       }
]]></script>
```

Този XML фрагмент би изглеждал обработен с парсера на Internet Explorer по следния начин (Фиг.6.6). Както се вижда, за по-добра четливост на документа, парсерът отделя данните в секцията CDATA с вертикална линия.



Фиг. 6.6 Представяне на CDATA секция с Internet Explorer

5. Стилове и таблици със стилове

Както беше показано до тук, XML акцентира върху структурата на данните, а не върху тяхното представяне. Но точно визуалното представяне на информацията е основната движеща тенденция в разработването на съвременните софтуерни продукти. XML, независимо дали е във формат за документ или за данни, се нуждае от някои допълнителни механизми за показване на информацията.

Когато се създава произволен таг в XML документ, въпросът който можем да си зададем е как браузерът ще разбере, как да изобрази данните в елемента на екрана. Отговорът не е утешителен – браузерът няма да знае как да изобрази информацията. Това всъщност е една от основните разлики между XML и HTML. HTML таговете имат определени значения и един HTML браузър обикновено притежава имплицитно дефинирана таблица със стилове за HTML таговете– конкретният формат може да е различен, но винаги има някакъв асоцииран стил.

Разликата между CSS в HTML и в XML е малка, така че голяма част от казаното за стиловете и таблиците със стилове в HTML може да се използва и тук. Една от основните разлики е, че атрибутът **style**, атрибутът **class**, както и елементите **<STYLE>** и **<LINK>** нямат никакво предварително определено значение. Поради тази причина, присвояването на стиловете се извършва посредством друг механизъм – инструкция за обработка в XML `<?xml-stylesheet ?>`.Както беше отбелязано, инструкцията за обработка е команда, предназначена за хост приложението (в този случай браузъра), което чрез нея трябва да извърши нещо с XML кода.

Инструкцията за обработка `<?xml-stylesheet ?>` работи с кой да е формат на таблица със стилове, който е дефиниран за XML, но в частност е приложим и за CSS. Пълният синтаксис на на командата за присвояване на стилове е следният:

```
<?xml-stylesheet type=mimetype href=stylesheetULR ?>
```

Синтаксисът е подобен на **<LINK>** елемента на HTML. Тук mimetype може да бъде препратка към който и да било език с таблици със стилове, който се подържа от браузъра, но обикновено стойността му е "**text/css**" или "**text/xsl**". Той трябва да бъде включен след XML декларацията `<?xml version='1.0'…?>`, но преди първият елемент от самия XML документ. Нека разгледаме един пример за създаване на външна таблица със стилове за XML документ, като за целта създадем един прост XML документ (notesTest.xml)

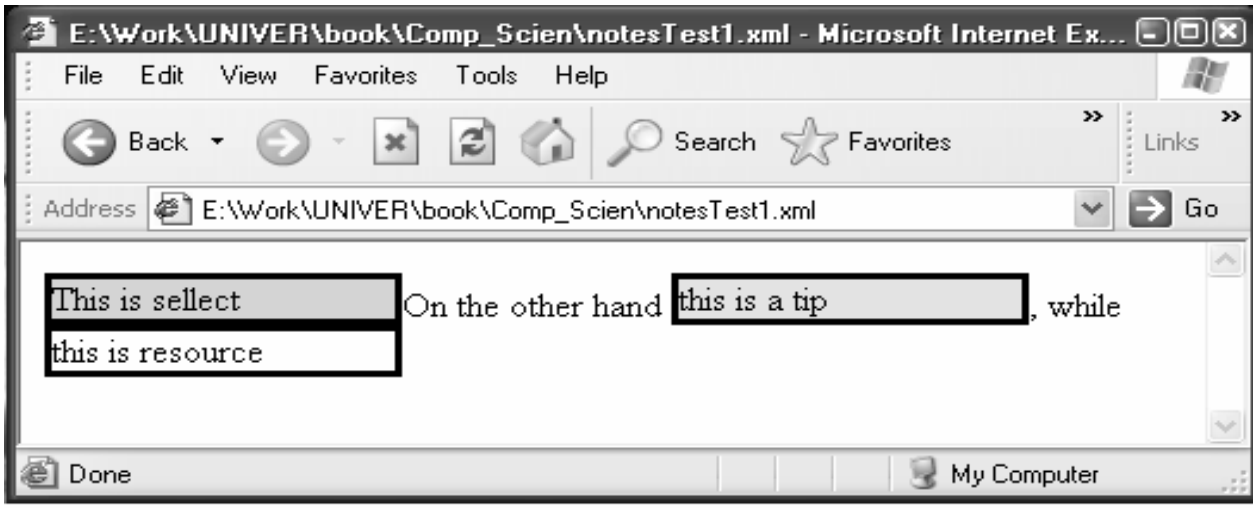
```
<?xml-stylesheet type="text/css" href="notes1.css"?>
<document>
     <body>
     <warning> This is warning. </warning>
On the other hand <tip> this is a tip</tip>,
while <resource >this is resource</resource>
     </body>
</document>
```

Този документ съдържа препратка към таблица със стилове **notes1.css**. Тя е почти идентична с HTML таблица със стилове. За XML документ, таблицата със стилове изглежда по следния начин:

```
tip{margin-left:.5;width:150px;border:solid 3px
black;position:relative; background-color:yellow;}
resource{margin-left:.5;width:150px;border:solid 3px
black;position:relative; background-color:white;}
sellect{margin-left:.5;width:150px;border:solid 3px
black;position:relative; background-color:pink;}
```

Ако се покаже XML документа в Internet Explorer, той би изглеждал, както това е показано на фиг. 6.6.

Това което прави впечатление при създаването на таблицата със стилове в разгледаният пример е, че много данни се повтарят за различните стилове (правила), които имат близки свойства. По полезно би било, ако общите данни се опишат в един блок, а различните свойства да се зададат като атрибути. Това в XML може да се постигне (ако браузерът подържа CSS2 спецификация), чрез създаване на атрибут от типа **type,** който систематизира всички близки по свойства стилове. Ето как изглежда таблицата със стилове при дефиниране на атрибут **type**:



Фиг. 6.6 Internet Explorer вид на XML документ с външна таблица на стилове

```
note {margin-left:.5in;width:150px;border-width:3px;
border-style:outset;border-color:blue; position:relative;
background-color:lightBlue;}
note[type=”typ”] {background-color:yellow; border-
color:yellow;}
note[type=”resource”] {background-color:green; border-
color:green;}
 note [type=”sellect{background-color:red; border-
color:red;}
```

За да се използва тази таблица със стилове изходният текст на XML документа трябва да се промени, като се използва таг <note> с атрибути:

```
<?xml-stylesheet type="text/css" href="notes2.css"?>
<document>
     <body>
     <note type="sellect"> This is sellect </sellect> On the
other hand <note="tip"> this is a tip
</tip>,while<note="resource">this is resource </resource>
     </body>
</document>
```

Въпреки, че външните таблици със стилове са много удобни, за поддържане на стилове в XML може да се използва задаване на стилове в самия документ. Както всички стандартни тагове от HTML, така и тага **<style>** няма никакво предварително зададено значение за XML. Следователно, таблиците със стилове не могат да се задават, както в HTML. В XML могат да се включат правила за таблици със стилове посредством инструкцията **xml-stylesheet**. Ето един пример за използване на вътрешни XML таблици с каскадни стилове.

```
<?xml-stylesheet type="text/css" href="#myStyleSheet"?>
<document>
     <st id="myStyleSheet">
     title{font-size:16pt;disply:block;}
     p{font-size:11pt;disply:block;}
     p:first-letter{float:left;font-size:24;}
     st (disply:none}
     b{font-weight:bold;}
     </st>
    <Title> CSS Style Sheet in XML document </Title>
```

```
        <body>
         <p> You can use <b>myStyleSheet</b> in XML dokuments
</p>
        </body>
    </document>
```

Таблиците със каскадни стилове използват за подреждане на информацията при изобразяването й, така наречения "кутиен" модел. Идеята на модела е проста: информацията се помества в наличното пространство на специфични контейнери, които се предоставят от приложението, което обработва XML или HTML файловете. Такива контейнери могат да бъдат или страници с фиксирани границ, панели във Web браузера или времеви граници за гласов текст. Начинът на разполагане на съдържанието на документа се определя от свойството **display**. То може да има стойности **inline, block** и **none**. Когато е зададено свойство **inline,** елементът се помества във възможно най-малкото пространство, а следващият елемент се помества непосредствено до предхождащият го. Ако стойността на **display** е **block**, то елемента заема цялото налично пространство по посока на разполагането: отгоре надолу и отляво надясно. Елементът остава невидим, ако стойността на свойството **display** е **none**.

6. Други възможности на XML

Досега бяха разгледани някои от основните правила за конструиране на XML документи, както и таблиците с каскадни стилове, които могат да се използват за показвана на XML документи във Web браузъри. Това е добър начин за използване на XML документи. Но XML е много повече от един инструмент за форматиране на текст. Освен това, файловете може да не са структурирани, съгласно изискванията на CSS (ако не са предназначени за показване на екрана). Тогава се налага трансформиране на XML документа в някакъв друг тип документ. За тази цел се използва език за трансформиране в произволен текстово базиран формат– **Extensible Style for Transformations (XSLT).** Този език е подкомпонент на един по-голям език XSL (Extensible StyleSheet Language), който се използва за създаване на таблици със стилове.

За да могат да се извършват преобразованията в даден XML документи е необходимо наличието на таблица със стилове, която съдържа инструкции как да се третират отделните елементи в документа и XSLT машина – софтуер, който изпълнява инструкциите в таблиците със стилове. Парсерите са тип обработващи програми на XML документи, но те проверяват само синтаксиса на документа, без да се интерпретира информацията. Те не използват таблици със стилове, докато XSLT машините работят само с шаблони (таблици със стилове).

XML е спецификация, която осигурява ефективен начин за обмяна на данни между хора и компютри. XML структурира данните и тази гарантирана структура позволява лесното предаване и трансформиране на данните. В това отношение XML може да взаимодейства със стандартните бази от данни и да осигурява доста ефективна комуникация между тях.

Съществуват различни типове бази данни – релационни, йерархични, обектно-ориентирани. Най-разпространеният тип са релационните бази от данни. Основата на такъв тип данни са таблиците, а за организиране на работата с таблици е разработен специален програмен език SQL (Structured Query Language). Интегрирането на XML структурирани данни и стандартните бази от данни е една от сериозните перспективи за използване на XML. XML може да се използва за връзка между различни типове бази от данни, както и за разработка на структури за обработка на данните. Това разбира се е по-сериозна материя и изисква специално разглеждане.