
Софийски университет “Св. Климент Охридски”
Факултет по математика и информатика

# XML – Информационни С-ми – курс 3

## л(1) Какво е XML?

СУ “св. Климент Охридски”, ФМИ,
Кат. Компютърна Информатика
доц. Й. Патиас
patias@fmi.uni-sofia.bg

# XMLтехнологии

Литература:
Програмиране с XML. Изд. СофтПрес, 2001

# Материал

- Глава 1, както е дадена в материала на Доц. П. Павлов „005 xml_course_1.ppt“

# Какво е XML?

Extensible Markup Language (XML) –
Разширяем Език за Форматиране (Маркиране)

Данни, файлове, текст

XML е технология, която се занимава с описанието и структурирането на данни.

Как компютрите съхраняват данните и извършват достъп до тях?

Двоични файлове - най-просто казано, един двоичен файл представлява поток от битове (нули и единици).

Работата на приложението, което е създало файла, е да разбира какво означават всички тези битове.

Когато една програма за текстообработка създаде документ, тя създава двоичен файл със свой собствен формат.

Програмата вмъква двоични кодове за отбелязване на удебелен текст, други кодове за нова страница и т.н.

Когато се отвори такъв документ в програмата за текстообработка, тя интерпретира тези кодове и показва правилно форматирания текст на екрана или го отпечатва на принтера.

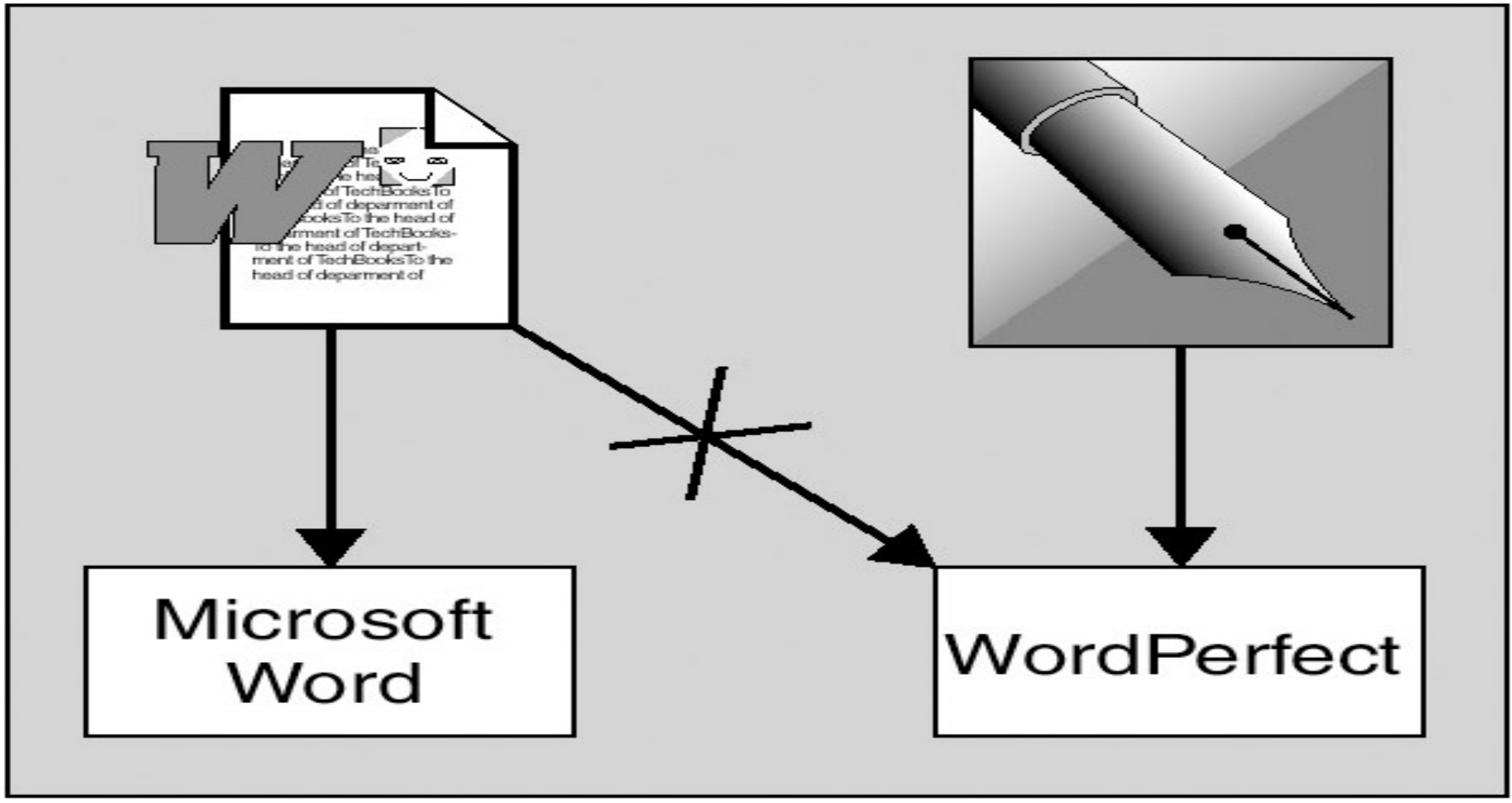
Вмъкнатите в документа кодове могат да се нарекат метаданни (metadata) или информация за информацията (например "тази дума трябва да е удебелена", "това изречение трябва да е в центъра на документа" и т.н.).

Точно в тези метаданни се крие разликата между различните типове файлове – различните типове файлове използват различни метаданни.

Един документ на програма за текстообработка има различни метаданни от този на програма за електронни таблици, тъй като те описват различни неща.

Предимството на двоичните файлови формати е, че компютрите лесно разбират двоичните кодове, което означава, че тези файлове се обработват
много по-бързо и са много ефикасни за съхраняването на метаданни.

Не може да се приема, че създаден от дадена програма за текстообработка документ ще се чете от друга подобна програма.



## Текстови файлове

Текстовите файлове също като двоичните файлове представляват поток от битове. Но във текстовите файлове тези битове са групирани заедно по стандартен начин (в байтове), така че винаги да представят числа. След това тези числа се съпоставят със знаци. Поради съществуването на подобни стандарти, текстовите файлове могат да се четат от много, много приложения, та дори и от хора посредством текстов редактор.

В началото Интернет беше почти изцяло текстово-базиран, което даваше възможност на хората да комуникират помежду си сравнително лесно. Това допринесе за експлозивното разпространение на Интернет и вездесъщността на негови приложения от рода на електронна поща, World Wide Web, нюзгрупи и т.н.

Недостатъците на текстовите файлове са, че в тях е по-трудно да се добавя друга информация - или казано с други думи, метаданни.

Повечето програми за текстообработка позволяват да се запазват документи в текстов формат, но в такъв случай не може да се форматира дадена секция с удебелен шрифт, или да се вмъкне двоичен файл за изображение. Ще може да се извличат само думи, но не и форматиращата информация.

# Кратка история на езиците за форматиране

Своите предимства имат както двоичните файлови формати (лесно разбираеми от компютъра, компактни), така и текстовите файлове (универсално заменяеми).

Нямаше ли да е най-добре да съществува един формат, който да комбинира универсалността на текстовите файлове с ефикасността и богатите възможности за съхраняване на двоичните файлове?

Идеята за универсален формат за данни не е нова. Всъщност откакто съществуват компютрите, програмистите се опитват да намерят начин за обмен на информация между различните програми. Един от първите опити да се комбинира един универсално заменим формат с богати възможности за съхраняване е SGML (Standard Generalized Markup Language).

SGML (Standard Generalized Markup Language).

Това е един текстово-базиран език, който може да се използва за форматиране (mark-up) на данни, т.е. за добавяне на метаданни, по един самоописателен
начин.

SGML беше предназначен да се използва като универсален начин за форматиране на данни за произволна цел и се прилагаше предимно в огромни системи за управление на документи. Оказва се, че когато стане въпрос за огромни количества сложни данни, възниква голямо количество съображения, в резултат на което SGML става много сложен език. Но със сложността идва и мощта.

Най-известното приложение на SGML е HyperText Markup Language  (HTML).

Тъй като правилата за създаване на SGML документи са толкова добри и SGML се е доказал в толкова много системи за управление на документи, беше напълно нормално да се създаде един специфичен речник HTML - който да се яви като универсален език за форматиране, който да служи за представянето на информация и за свързването на различни порции информация. Идеята беше всеки HTML документ (или Web страница) да може да се представи във всяко приложение, което разбира HTML (така наречения браузър).

Браузърът трябва да може не само да показва документа, но и ако документът съдържа хипервръзки към други документи, да може да ги покаже и тях.
Нещо повече, понеже HTML е текстово-базиран, то всеки може да създаде HTML страница с помощта на прост текстов редактор или с някоя от многобройните програми за редактиране на Web страници. Всеки HTML редактор, включително текстов редактор, може да създаде един HTML файл, като този файл може да бъде разгледан във всеки Web браузър в Интернет.

# Какво все пак е XML?

SGML е много сложен език и не е пригоден за обмен на данни през Web.

HTML има ограничения – само показва документи в браузър.

Може да се създаде HTML документ, който да показва данните за един човек, но това е всичко, което може да се прави с този документ.
От него например не може да се разбере коя част от информацията се отнася за първото име на човека, защото HTML няма средства за описанието на подобна специализирана информация.

Поради тази причина е създаден Extensible Markup Language (XML).

XML е подмножество на SGML, от което са премахнати най-усложняващите неща.

XML е напълно съвместим с SGML. Обратното не е вярно.

Все пак е важно да се разбере, че XML всъщност не е "език", а стандарт за създаване на езици, които отговарят на XML критерии.

Казано с други думи, XML описва синтаксиса, който да се използва за създаване на свои собствени езици.

Например да предположим, че имаме данни за едно име и искаме да можем да споделяме тази информация с други хора.

Но освен това искаме да можем да използваме тази информация и в компютърна програма. Вместо просто да създадем следния текстов файл:

John Doe

или HTML файл:

```
<HTML>
<HEAD><TITLE>Name</TITLE></HEAD>
<BODY>
<P>John Doe</P>
</BODY>
</HTML>
```

можем да създадем един XML файл:

```
<name>
    <first>John</first>
    <last>Doe</last>
</name>
```

От този прост пример може да се види защо езиците за форматиране от рода на HTML и XML се наричат самоописателни. Гледайки данните, може лесно да се каже, че тази информация се отнася за <name> (име). Има данни, наречени <first> (първо), и други данни, наречени <last> (последно).

(Бихме могли да наречем данните както поискаме. Добре е да използваме избора правилно и да даваме на нещата имена, които имат смисъл.)

XML версията на тази информация е много по-голяма от тази с обикновен текст.

Използването на XML за форматиране на данни увеличава техния размер, при това понякога значително, но малкият размер на файловете не е една от основните цели на XML.

Той има за цел да улесни писането на софтуер, който извършва достъп до данни, като задава структура на данните.

По-големият размер на файловете не трябва да възпира от използване на XML. Предимствата от лесен за писане код са много по-големи, отколкото недостатъците от увеличаване на размера на данните за предаване.

Ако размерът е критичен за приложенията, то винаги може да се компресират XML файловете, преди да се предадат по мрежата - компресирането на текстови файлове дава много добри резултати.

# Отваряне на XML файл в браузър

Вградена таблица със стилове (style sheet), която се прилага за подразбиращото се форматиране на всеки XML документ.

Комбиниране на XML данни с таблици със стилове -> разделяне на данните от тяхното представяне.

С XML съществува стандартизиран начин за извличане на необходимата ни информация независимо от това, как е структурирана.

Колкото по-сложни са данните, с които трябва да се работи, толкова по-сложна ще става и логиката, която трябва да се реализира.

Точно в големите приложения може да се оцени най-добре XML.

# XML парсери (синтактични анализатори)

Има програми, наречени парсери, които могат да четат XML текста и да извличат информацията вместо нас. Можем да използваме тези парсери в нашите собствени програми, което означава, че нашите приложения никога няма да работят с XML директно.

Можем да подадем на XML парсера подобен файл:

```
<name>
    <first>John</first>
    <middle>Fitzgerald Johansen</middle>
    <last>Doe</last>
</name>
```

и той ще ни каже, че има една част от данните, наречена <middle>, а информацията, която се съхранява в нея, е Fitzgerald Johansen.

Създателят на парсера не е знаел никакви правила за това, къде свършва малкото име и започва презимето. Той не е знаел нищо нито за приложението, нито за <name>. Езикът, на който е написан XML кода, също няма никакво значение за парсера. XML код, написан на английски, китайски или който и да било друг език, може да се прочете от същия парсер независимо от това, че човекът, написал парсера, може да не е знаел нито един от тези езици.

Има още едно допълнително предимство: ако по-рано е написана програма за работа с XML формат, в който има само име и фамилия, тя ще може да работи и с новия XML формат, без да се налага да се променя кода. Това е така, защото парсерът извършва работата по извличането на данните от документа, а ние можем да добавяме нови неща към нашия XML формат, без да разваляме кода, така че ако желаят, новите приложения могат да се възползват от добавената информация. (Ако премахнем елементи от нашия пример с <name> или променим техните имена, ще трябва да променим и кода в нашите приложения, така че да работи с модифицираните имена).

Има още едно допълнително предимство: ако по-рано е написана програма за работа с XML формат, в който има само име и фамилия, тя ще може да работи и с новия XML формат, без да се налага да се променя кода.

Това е така, защото парсерът извършва работата по извличането на данните от документа, а ние можем да добавяме нови неща към нашия XML формат, без да разваляме кода, така че ако желаят, новите приложения могат да се възползват от добавената информация. (Ако премахнем елементи от нашия пример с <name> или променим техните имена, ще трябва да променим и кода в нашите приложения, така че да работи с модифицираните имена).

# Защо *"Extensible"?*

Тъй като имаме пълен контрол над създаването на нашия XML документ, то можем да групираме данните както си пожелаем, или както е удобно за нашето конкретно приложение. Ако не се нуждаем от гъвкавостта на примера с <name> и решим да опишем името по следния начин:
<designation>John Fitzgerald Johansen Doe</designation>
можем спокойно да го направим.

Ако искаме да създадем такива данни, които да могат да бъдат използвани само от една конкретна програма, няма проблем. Ако искаме да споделим нашите данни с други програми или дори други компании в Интернет, XML ни дава нужната гъвкавост. Свободно можем да структурираме едни и същи данни по различен начин, който да е подходящ за дадено приложение или за цяла група от приложения.

Ето откъде идва думата *"Extensible"* (разширяем) в името на езика Extensible Markup Language: с помощта на този език всеки може да форматира данните по произволен начин, дори другите да го правят по коренно различни начини.

Разбира се, предимствата на XML се виждат по-ясно, ако хората използват един и същ формат за общите дейности, тъй като това ни позволява да обменяме информация по-лесно. Вече съществуват множество проекти (и реализации) за създаването на стандартни за индустрията речници (vocabularies) за описването на разнообразни типове данни.

Например:
Scalable Vector Graphics (SVG) е XML речник за описание на двумерни графики,
MathML е XML речник за описание на математически операции като база за комуникация от тип машина с машина,
Chemical Markup Language (CML) е XML речник за управление на химическа информация.

Имате възможност да напишете свои собствени XML речници за описание на типовете информация, които ви трябват, но ако използвате по-общи формати имате по-голям шанс да създадете софтуер, който е съвместим с други приложения.

Ако създадем един XML документ, можем да сме сигурни,че всеки XML парсер ще извлече информацията от този документ, но не можем да гарантираме, че всяко приложение ще разбере какво означава тази информация. Това е така, защото парсерът може да ни каже, че има една част от данните, наречена <middle> и информацията, която се съхранява в нея, е "Fitzgerald Johansen". Това обаче не означава, че в света съществува софтуер, който знае какво е <middle>, за какво се използва или какво означава.

Следователно можем да създаваме XML документи за описание на всяка информация, която ни трябва, но преди да можем да сметнем XML за полезен, трябва да има написани приложения, които да го разбират.

# Йерархии с информация

Как са структурирани данните в един XML документ?

Когато се работи с големи или средни количества информация, обикновено е по-добре те да бъдат групирани в свързани подтеми, вместо цялата информация да се представя като едно голямо кълбо.

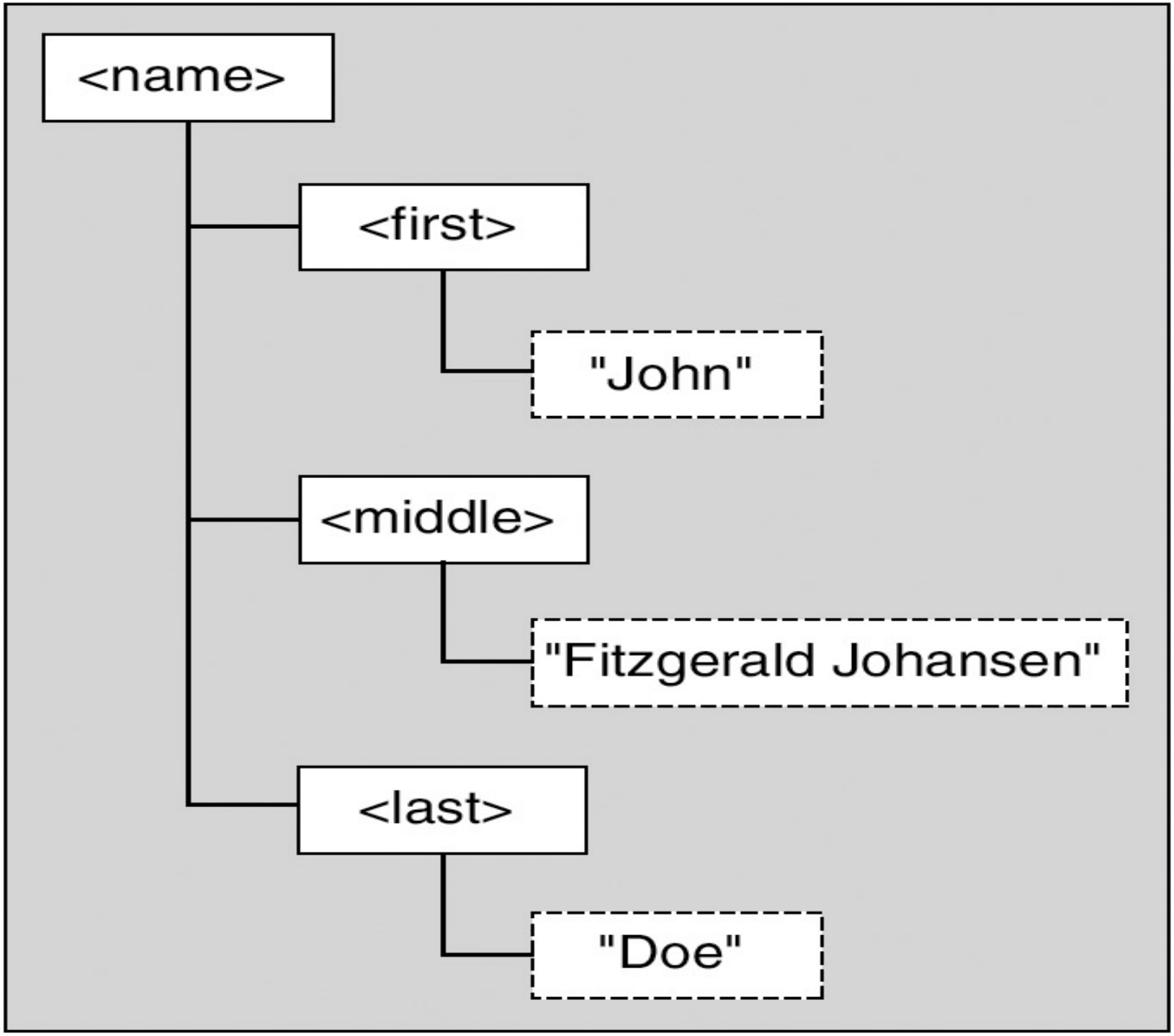
Например една глава от книга е разделена на подтеми, които от своя страна са разделени на параграфи, а те на изречения и т.н. Тази йерархия улеснява разбирането на информацията, както и достъпа до нея.

# Йерархия в XML

XML също групира информацията в йерархии.

Взаимовръзките между нещата (елементите) в нашите документи са от тип родител/наследник или брат/сестра.

Нека да разгледаме нашия пример с <name>, показан йерархично:


<name>
<first>
"John"
<middle>
"Fitzgerald Johansen"
<last>
"Doe"

<name> е родител на <first>, който е наследник (дъщерен елемент) на <name>. <first>, <middle> и <last> са братя и сестри един спрямо друг (всички те са наследници на <name>). Ще обърнем внимание, че текстът е наследник на съответния елемент. Например "John" е наследник на <first>.

Тази структура се нарича също дърво. Всяка част от дървото, която съдържа наследници, се нарича клон, докато частите, които нямат наследници, се наричат листа.

Понеже елементът <name> има за наследници само други елементи, то се казва, че той има елементно съдържание **(element content).**

Обратно, <first>, <middle> и <last> имат само текст като наследници, така че се казва, че те имат просто съдържание **(simple content).**

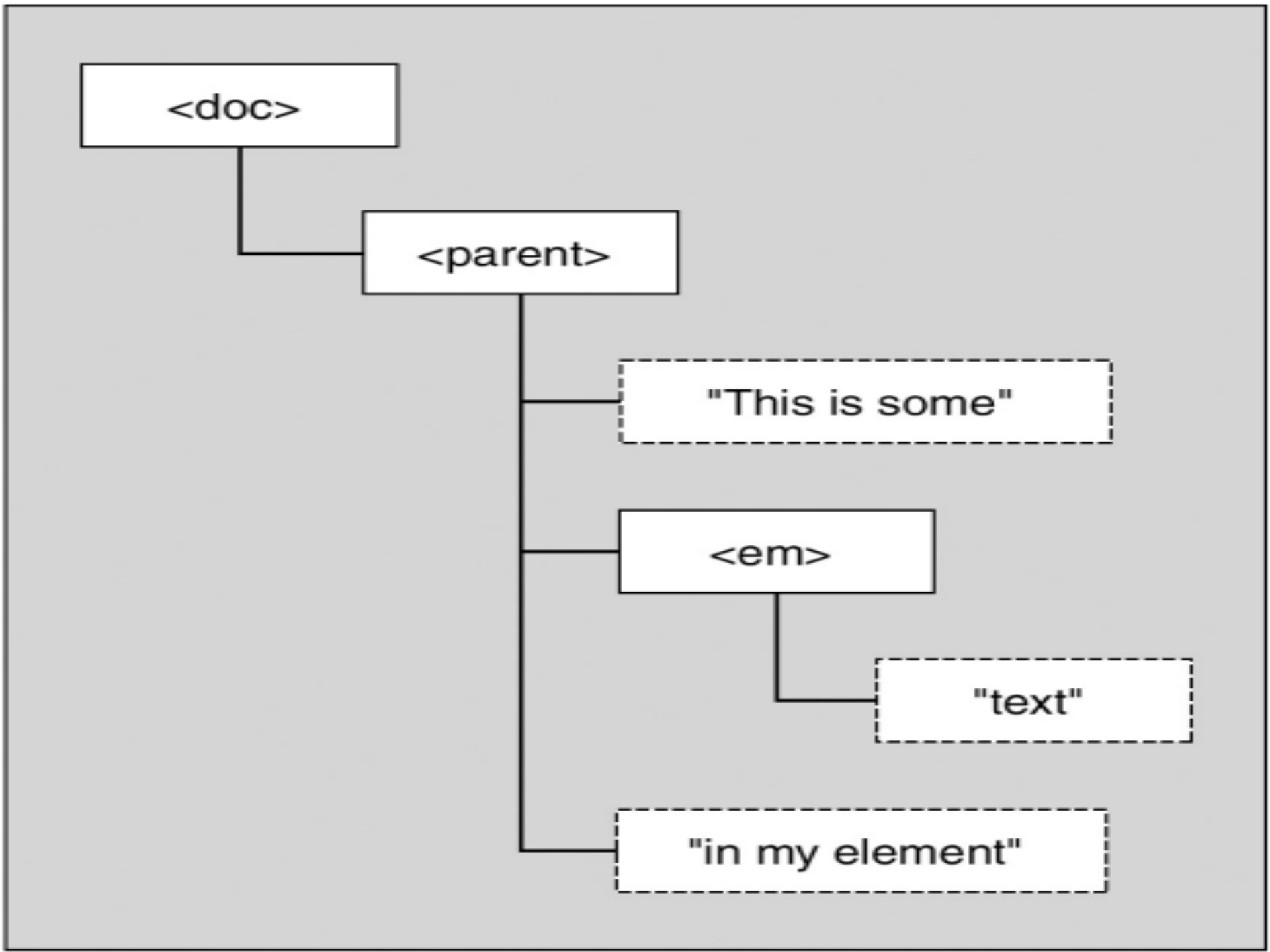
Елементите могат да съдържат както текст, така и други елементи. За тях се казва, че имат смесено съдържание **(mixed content).**

Например:

```
<doc>
    <parent>this is some <em>text</em> in my element</parent>
</doc>
```

Тук елементът <parent> има три наследника:
- текстов наследник, съдържащ текста "this is some"
- наследник 
- друг текстов наследник, съдържащ текста "in my element“

Структурата на документа е следната:


<doc>
<parent>
"This is some"

"text"
"in my element"

Взаимовръзката може да се дефинира с помощта на аналогията на родословното дърво:

- <doc> е предшественик на ;
-  е потомък на <doc>.

# Какво е документен тип?

Красотата на XML е в неговата способност да описва произволна информация. Той е безкрайно гъвкав по отношение на това, как структурираме нашите данни. Но евентуално може да поискаме да установим определен дизайн за нашата информация и да кажем "за да се спазва нашият XML формат, данните трябва да са структурирани по следния начин".

Например, когато създадохме по-горния XML елемент <name>, ние конструирахме някакви структуриращи данни. Не само че включихме цялата информация за едно име, но нашата йерархия съдържа също така и неявна информация за взаимовръзката между някои от частите с данни (например елементът <name> съдържа един елемент <first>.

Но има и нещо повече.
Създадохме и специфичен набор от елементи, който се нарича речник. Т.е. ние сме дефинирали няколко XML елемента, които съвместно формират едно име: <name>, <first>, <middle> и <last>.

Най-важното е, че сме създали един документен тип. Създали сме специфичен тип документ, който трябва да е структуриран по определен начин, за да опише специфичен тип информация. Въпреки че не сме ги дефинирали явно, съществуват няколко правила, които трябва да бъдат съблюдавани от елементите на нашия речник, за да отговаря документът <name> на нашия документен тип.

Например:

- Най-горният елемент трябва да е елементът <name>.
- Елементите <first>, <middle> и <last> трябва да са наследници на този елемент.
- Елементите <first>, <middle> и <last> трябва да са подредени по този начин.
- В елементите<first> и <last> трябва да има информация, но за елемента <middle> това не е задължително.

Съществуват различни видове синтаксис, които могат да се използват за формалното дефиниране на един XML документен тип. Някои XML парсери знаят как да четат този синтаксис и могат да го използват, за да определят дали един XML документ действително отговаря на правилата на документния тип или не.

Всички използвани дотук видове синтаксис за дефиниране на документен тип имат недостатъци. Те могат да осигуряват някаква проверка за типове, но за много приложения тя е недостатъчна.
Те не могат да изразят човешкото значение на термините в речника.

Поради тази причина, когато се създават XML документни типове, трябва да се предостави и документация за тях. Например, ако искаме и другите да могат да използват формата от примера <name> в своите XML документи, трябва да им предоставим документация, описваща как работи нашият формат.

# Какво е документен тип?

Нарекохме нашия <name> пример документен тип.
Другите, които работят с XML, могат да го нарекат и по друг начин.

Когато се използва XML за създаването на документни типове не трябва да се мисли за това, че се създават документни типове - просто да се проектира XML по начин, който има смисъл за приложението и да се използва. Тук са избрани термините “документен тип” и “речник”, защото вършат добра работа за описването на това, което трябва да се опише. Те не са универсални типове, използвани от всички. По-важни са концепциите, независимо от термините, които се използват.

Една от причините, поради които HTML и XML са толкова добри идеи, е, че те са стандарти. Това означава, че всеки може да следва тези стандарти и решенията, които разработва, ще могат да си взаимодействат.

# От какво е изграден XML?

Съществуват множество свързани помежду си спецификации, които съвместно формират XML семейството от технологии, като всяка спецификация покрива различен аспект от комуникирането на информация. Ето някои от най-важните:

XML 1.0 е базовата спецификация, върху която е изградено XML семейството. Тя описва синтаксиса, който XML документите и XML парсерите трябва да следват, както и всичко друго, което ви е необходимо за четенето или писането на XML документ.

Тъй като можем да създаваме свои собствени структури и имена на елементи в нашите документи, DTD дефинициите (Document Type Definition) и Схемите (Schemas) ни осигуряват начини за създаването на шаблони за нашите документни типове. Можем да проверяваме дали други документи са съвместими с тези шаблони, а освен това и други разработчици могат да създават съвместими документи.

Пространствата от имена (Namespaces) ни дават средства, чрез които да отличаваме един XML речник от друг. Това ни позволява да създаваме по-богати документи, като комбинираме множество речници в един документен тип.

XPath описва език за заявки, чрез който се обръщаме към части от един XML документ. Това позволява на приложенията да искат специфична част от даден XML документ, а не винаги да работят с цялата информация.

Например XPath може да се използва, за да се вземат "всички фамилии" (фамилни имена) от даден документ.

В някои случаи ще искаме да показваме нашите XML документи.

В по-простите случаи може да използваме таблици с каскадни стилове (Cascading Style Sheets - CSS), за да дефинираме представянето на нашия документ, а в по-сложните Extensible Style Sheet Language (XSLT), който може да трансформира нашите документи от един тип в друг, както и обекти за форматиране (Formatting Objects), които се занимават с показването.

XPointer и XLink представляват езици, които се използват за свързването на XML документи по подобен на HTML хипервръзките начин.

За да се осигури на по-традиционните приложения интерфейс към XML документи, съществува един документен обектен модел (Document Object Model - DOM).

Къде се използва XML и къде може да бъде използван?
XML не зависи от използваните платформа и програмен език, което значи, че това няма значение за компютрите, с които се работи.

Следват няколко примера за използване на XML:

## Намаляване на товара върху сървъра

Web-базираните приложения могат да използват XML за намаляването на натоварването върху Web сървърите. Това може да се постигне, като цялата информация се пази в клиента възможно най-дълго, след което се изпраща на сървърите в един голям XML документ.

# Съдържание на Web сайт

XML се използва за писането на спецификации. Тези XML документи могат след това да бъдат трансформирани в HTML (чрез XSLT) или в някой друг от множеството формати за представяне на данни.
Някои Web сайтове също използват вътрешно XML за своето съдържание там, където по традиция се използва HTML. Този XML може след това да бъдат трансформирани в HTML (чрез XSLT) или може да бъде показан директно в браузъра посредством CSS.

В действителност Web сървърите могат дори динамично да определят какъв вид браузър извлича информацията, след което да решат какво да правят (например да трансформират XML кода в HTML за по-старите браузъри, или да изпратят XML кода направо на клиента, като по този начин намалят товара върху Web сървъра, в случай на на нов браузър).

# Електронна търговия

Електронна търговия (e-commerce) е нещо, което се говори постоянно в последно време. Компаниите откриха, че комуникацията посредством Интернет, вместо чрез по-традиционните способи (факс, човек с човек и т.н.), може да рационализира техните дейности, като намали разходите и увеличи времето за реагиране. XML е перфектният формат за обмен на данни, който една компания може да използва, за да изпрати информация на друга компания.

Положението, при което дадени компании имат някакъв вид текуща взаимовръзка, е известно като бизнес-към-бизнес (business to business - B2B) електронна търговия. Освен това съществуват и бизнес-към-консуматор (business to consumer - B2C) транзакции. XML има своите приложения и в двата вида електронна търговия.

Освен това съществуват още множество приложения, в които е бил използван XML. Да се надяваме, че след като изслушате този курс, ще сте в състояние да решите къде може да използвате XML и къде не.