SGML, XML

Igor Wojnicki

Katedra Informatyki Stosowanej
Akademia Górniczo-Hutnicza w Krakowie

22 stycznia 2014

Spis Treści

Spis Treści

Standard SGML

- SGML (ang. *Standard Generalized Markup Language*)
- powstał do zastosowań przemysłowych (formalizacja dokumentów prawnych) (1969)
- znalazł liczne zastosowania (wojsko, NATO, wydawnictwa)
- pozwala na definiowanie składni i semantyki języków
- w pełni uniwersalny
- definiuje się w nim HTML (walidacja!)
- umożliwia opis *dowolnych* języków znaczników – *metajęzyk*
- SGML jest *starszy* od HTML, XML, W3C, WWW, Internet

Czym jest SGML

- standaryzowany *metajęzyk*, służy do tworzenia innych języków
- norma ISO: *ISO 8879:1986*
- pozwala na definiowanie języków opisujących strukturę dowolnie złożonych dokumentów
- zawiera mechanizmy badania poprawności składniowej
- umożliwia praktycznie dowolne konwersje dokumentów (dodatkowe pakiety)
- stał się niekwestionowanym standardem przemysłowym

DTD i walidacja

- język w SGML ma *formalnie* zdefiniowaną gramatykę
- definicja gramatyki języka zawarta jest w DTD
- DTD = *Document Type Definition*
- DTD jest zapisana w EBNF
- parser SGML pozwala na przeprowadzenie weryfkiacji składniowej dokumentu („walidacji”)
- bezbłędna walidacja oznacza poprawność *składniową* dokumentu
- DTD jest *konieczne* przy pracy z SGML

Spis Treści

Aplikacje SGML

- język opisany w SGML przy pomocy DTD nazywa się *aplikacją* SGML
- DTD zawiera elementy i atrybuty, co pozwala na odpowiednie użycie znaczników
- DTD jest zawsze *kluczowe* dla aplikacji SGML
- zmiany w DTD wymuszają konwersję dokumentów aplikacji
- przy pomocy SGML można zdefiniować HTML

Rozwój, zastosowania, ogranicznia

- SGML jest możliwie najogólniejszym językiem do tworzenia języków znaczników
- zastosowania w systemach tworzenia dokumentacji (DocBook) katalogowania dla przemysłu, dokumentów prawnych, itd.
- niezwykle rozbudowany – trudności z implementacją narzędzi
- parsing uzależniony od DTD

Przykład I

```
<!doctype linuxdoc system>

<!-- Here's an SGML example file
 -->

<article>
<title>Quick SGML Example
<author>Matt Welsh, <tt/mdw@cs.cornell.edu/
<date>v1.0, 28 March 1994
<abstract>
This document is a brief example using the Linuxdoc-SGML DT
</abstract>

<toc>

<sect>Introduction
```

Przykład II

```
<p>
This is an SGML example file using the Linuxdoc-SGML DTD.
You can format it
using the command
<tscreen><verb>
% sgml2txt example.sgml
</verb></tscreen>
this will produce plain ASCII. You can also produce LaTeX,
and GNU info.

<sect>The source

<p>
Looking at the source for this file will
be instructive to show you how
to use many of the Linuxdoc-SGML constructs.
```

Przykład III

```
You should also read the
<em/Linuxdoc-SGML User's Guide/,
in the file <tt/guide.sgml/.

The source looks and feels like LaTeX,
as you can see. Paragraphs are
separated by blank lines,
macros are enclosed in angle brackets.
It's quite simple.

<sect>Some examples

<sect1>Lists

<p>
Lists are easy as well.
Just use the <tt/itemize/ element with the
```

Przykład IV

```
<tt/item/ commands, seen here:
<itemize>
<item> This is a list.
<item> Nothing exciting about that.
<item> A final item to top it all off.
</itemize>

</article>
```

Spis Treści

Narodziny XML

- XML (ang. *eXtensible Markup Language*)
- podzbiór SGML (XML *NIE* zastępuje SGML!)
- zorientowany na sieć – integracja z protokołami
- od początku standaryzowany przez W3C
- rozszerzalny i modularny (a nie bardzo złożony)
- łatwy w nauce, użyciu i implementacji narzędzi
- XML to *metajęzyk*! (jak SGML)

Rozwój XML

1. *Extensible Markup Language (XML) 1.0 W3C Recommendation*, 10.02.1998 `www.w3.org/TR/1998/REC-xml-19980210`
2. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, 26.11.2008, `http://www.w3.org/TR/2008/REC-xml-20081126/`,
3. *Extensible Markup Language (XML) 1.1* aktualna specyfikacja `www.w3.org/TR/xml11`

Przykład XML

```
<?xml version="1.0"?>
<ala>
    <ma>kota</ma>
    <kot>ma</kot>
    ale
</ala>
```

Założenia XML

- integracja z Internetem
- uniwersalny, wiele zastosowań
- kompatybilny z SGML
- łatwe pisanie narzędzi przetwarzających
- zredukowane możliwości na poziomie języka
- rozszerzalność
- dokumenty można czytać bez narzędzi
- możliwości walidacji (przy pomocy DTD)
- migracja z HTML

Podstawowy dokument

```
<?xml version="1.0"?>
<greeting>
    <hello>Hello World!</hello>
</greeting>
```

Składa się *koniecznie* z *deklaracji* (*prologu*) i *jednego* elementu głównego.

Znaczniki

- znaczniki mają postać `<nazwa>`
- każdemu znacznikowi otwierającemu odpowiada zamykający `</nazwa>`
- znacznik pusty `<nazwa/>` (`<nazwa></nazwa>`)
- znaczniki są wrażliwe na wielkość liter
- znaczniki można wielokrotnie zagnieżdżać, pamietając o kolejności zamykania
- struktura drzewiasta!
- nie ma *żadnych* predefiniowanych znaczników!

Atrybuty

- atrybuty mogą dookreślać cechy obiektów opisanych znacznikami
- atrybuty umieszcza się *tylko* w znaczniku otwierającym
- składnia:

```
<znacznik atr1="wartA" atr2="wartB">
```

- atrybuty mogą być: tekstowe (`cdata`), atomiczne (tokeny, identyfikatory), wyliczeniowe (predefiniowane zakresy)
- atrybuty mogą być opcjonalne (`implied`), obowiązkowe (`required`) (ale z DTD!), domyślne (`"wart"`), domyślne ustawione (`# FIXED "wart"`)

Encje

- *encja* to obiekt przechowywania informacji
- dokument XML ma zawsze przynajmniej jedną encję (*document entity*)
- encje moga być wewnętrzne (znakowe) i zewnętrzne (odnoszą się do innych plików)
- encje trzeba deklarować:

```
<!ENTITY nazwa "tresc">
```

- zbiór encji tworzy strukturę *fizyczną* dokumentu

Encje

- encje deklaruje się w DTD:

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE cds [
    <!ENTITY encja "nasza pierwsza">
]>
```

- odwołanie do encji `&nazwa;`
- jest dokładnie 5 encji predefiniowanych:

```
&lt; &gt; &amp;  &apos; &quot;
```

XML a SGML i HTML

- język XML jest podzbiorem SGML
- XML ma mniej opcji, ale jest rozszerzalny
- pozwala na walidację, lecz nie wymusza jej
- XML i SGML to *metajęzyki*
- HTML to konkretny język przedmiotowy
- HTML (w różnych wersjach DTD) jest definiowalny jako aplikacja SGML
- przy pomocy XML jest zdefiniowany język XHTML odpowiadający HTML 4

Spis Treści

Struktura fizyczna

- jest związana z encjami: parsowane, nie parsowane
- dokument z fizycznego p. widzenia składa się z szeregu encji
- encje moga się odwoływać do kolejnych encji w innych plikach
- z fizycznego p. widzenia dokument może składać się z wielu plików
- granice encji nie muszą się pokrywać z granicami elementów
- struktura fizyczna – płaska

Struktura logiczna

- z logicznego punktu widzenia dokument XML ma strukturę drzewa
- jest ona wyznaczana przez kolejne, następujące po sobie i zawierające się w sobie elementy
- drzewo ma zawsze jeden korzeń, 1. el. dokumentu
- korzeń pokrywa się z 1 (często jedyną) encją
- prawidłowo skonstruowane dokumenty powinny mieć zsynchronizowane struktury logiczną i fizyczną
- `standalone`, do przetwarzania dokumentu nie są potrzebne żadne zewnętrzne definicje (znaczników, atrybutów, itp.; zewnętrzne DTD)
- struktura logiczna – drzewiasta

Przestrzenie nazw

- elementy mają ustalone znaczenie jedynie wewnątrz konkretnego dokumentu
- oby powiązać elementy jednego dokumentu z innym mozna użyć przestrzeni nazw (ang. *namespace*)
- w tym celu trzeba zadeklarować przestrzeń:

```
<nazwaelem xmlns:prefix="URLprzestrz">
```

- następnie używa się ich:

```
<prefix:element>tresc</prefix:element>
```

Przestrzenie nazw

- domyślną przestrzeń (nie wymagająca prefiksu przy elementach) należy podać w pierwszym elemencie:

```
<nazwaelem xmlns="URLprzestrz">
```

Przestrzenie nazw, przykład I

```
<html xmlns:xdc="http://www.xml.com/books"
        xmlns="http://www.w3.org/HTML/1998/html4">
 <head><title>Book Review</title></head>
 <body>
  <xdc:bookreview>
   <xdc:title>XML: A Primer</xdc:title>
   <table>
    <tr>
     <td>Author</td><td>Price</td>
     <td>Pages</td><td>Date</td></tr>
    <tr>
     <td><xdc:author>Simon St. Laurent</xdc:author></td>
     <td><xdc:price>31.98</xdc:price></td>
     <td><xdc:pages>352</xdc:pages></td>
     <td><xdc:date>1998/01</xdc:date></td>
    </tr>
```

Przestrzenie nazw, przykład II

```
    </table>
   </xdc:bookreview>
  </body>
</html>
```

Przestrzenie nazw, przykład III

```
<h:html xmlns:xdc="http://www.xml.com/books"
        xmlns:h="http://www.w3.org/HTML/1998/html4">
 <h:head><h:title>Book Review</h:title></h:head>
 <h:body>
  <xdc:bookreview>
   <xdc:title>XML: A Primer</xdc:title>
   <h:table>
    <h:tr>
     <h:td>Author</h:td><h:td>Price</h:td>
     <h:td>Pages</h:td><h:td>Date</h:td></h:tr>
    <h:tr>
     <h:td><xdc:author>Simon St. Laurent</xdc:author></h:td
     <h:td><xdc:price>31.98</xdc:price></h:td>
     <h:td><xdc:pages>352</xdc:pages></h:td>
     <h:td><xdc:date>1998/01</xdc:date></h:td>
    </h:tr>
   </h:table>
```

Przestrzenie nazw, przykład IV

```
    </xdc:bookreview>
  </h:body>
</h:html>
```

Przestrzenie nazw, przykład V

```
...
<mytrades:portfolio xmlns:mytrades=
 "http://www.somedomain.com/ns/mytrades/">
  <mytrades:stock>Cisco</mytrades:stock>
  <mytrades:stock>Nortel</mytrades:stock>
  <mytrades:stock>eToys</mytrades:stock>
  <mytrades:stock>IBM</mytrades:stock>
</mytrades:portfolio>
...
```

Przestrzenie nazw vs. DTD

- Walidacja dokumentu jest problematyczna: 1 DTD, 1+ przestrzeni nazw
- Rozwiazania:
  - nie używać walidacji
  - skonstruować DTD zawierające wszystkie znaczniki z używanych przestrzeni nazw wraz z prefiksami!!!

Części CDATA

- te części dokumentu pozwalają na dokładne cytowanie tekstu
- powodują wyłączenie interpretowania XML
- są przydatne do wstawiana danych, w tym kodu w innym języku
- sposób użycia to:

```
<![CDATA[
    oto &jest; cos
    <hej><nie>inter</hej>pretowanego
]]>
```

Komentarze

- w XML można wstawiać komentarze
- komentarze są umieszczane tak:

```
<!-- komentarz -->
```

- komentarze nie mogą zawierać sekwencji `--`
- komentarze nie mogą być wewnątrz znaczników

Kodowanie znaków

```
<?xml version="1.0"?>
<?xml version="1.0" encoding="iso-8859-1">
<?xml version="1.0" encoding="iso-8859-2">
<?xml version="1.0" encoding="utf-8">
<?xml version="1.0" encoding="utf-16">
```

Parsery („procesory”) XML mają wspierać UTF, deklaracja w ASCII.
Metody autodetekcji:
`http://www.w3.org/TR/REC-xml/#sec-guessing`

Sterowanie przetwarzaniem

- *processing instructions* mają postać:

```
<?nazwa parametry ?>
```

- na przykład:

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

- działają podobnie do `CDATA`, lecz moga być wykorzystywane przez parser XML, ew. dodatkowe aplikacje
- p.i., których nazwy zaczynają się od `xml` są używane przez parsery XML

Spis Treści

Poprawność – po co?

- podobnie jak SGML, XML pozwala na określenie *poprawności* dokumentu
- *poprawność* ma charakter składniowy
- XML dopuszcza 2 poziomy poprawności
- badanie poprawności jest przydatne dla tworzenia bezbłędnych dokumentów
- w niektórych zastosowaniach poprawność może być niezbędna

Dobre sformułowanie/Poprawność składniowa

Dokument powinien być przynajmniej *dobrze sformułowany* (ang. *well formed*):

1. zawiera *przynajmniej* 1 el. fiz. i
2. zawiera *dokładnie* 1 el. log.
3. el. logiczne są prawidłowo zagnieżdżone
4. nazwy elementów są takie same w znacznikach otwierających i zamykających
5. nazwy atrybutów w danym elemencie nie powtarzają się, a wartości są cytowane
6. encje są deklarowane przed użyciem
7. wartości atrybutów nie odwołują się do zewnętrzych encji

Poprawność strukturalna

Aby można mówić o *poprawnym strukturalnie* dokumencie (ang. *valid*), *musi* być DTD!
Dokument jest *poprawny strukturalnie* jeżeli:

1. jest dobrze sformułowany
2. zawiera prawidłową deklarację typu dokumentu (odwołanie do DTD)
3. jest zgodny z DTD

Walidacja

- *walidacja* jest procesem sprawdzania poprawności
- jest przeprowadzana przez *parser XML*, który jest osobną aplikacją
- istnieje wiele parserów, walidujące, pozwalające na przetwarzanie dokumentu
- można również używać parserów SGML
- najbardziej znanym parserem XML jest *Expat* autorstwa J. Clarka (`www.jclark.com/xml`)
- najbardziej znanym parserem SGML jest *SP* autorstwa Jamesa Clarka (`www.jclark.com/sgml`, `opensp.sf.net`)
- sieciowy walidator: `validator.w3.org` (oparty o SP)

Przykład pliku XML

```
<?xml version="1.0" standalone="yes"?>
<cds>
  <cd>
   <title>Cryptic Writings</title>
   <band>Megadeth</band>
    Absolutely &nothing; we trust
  </cd>
</cds>
```

Przykład OpenSP

```
onsgmls -wno-valid -s xml.dcl t8.xml
onsgmls:t8.xml:6:16:W:
  cannot generate system identifier
  for general entity "nothing"
onsgmls:t8.xml:6:23:E:
  reference to entity "nothing" for
  which no system identifier
  could be generated
onsgmls:t8.xml:6:15:
  entity was defined here
```

Spis Treści

Gramatyki formalne i E/BNF

- istnieją różne metody formalizacji składni języka, na przykład wyrażenia regularne i gramatyki
- gramatyki generacyjne, redukcyjne
- gramatyki można formalizować przy pomocy E/BNF (ang. *Extended/Backus-Naur Form*)
- zapis gramatyki w postaci BNF ma postać szeregu *reguł produkcji*
- każda reguła opisuje fragment gramatyki:

```
<s_metajezyka> ::= <s_metajezyka>
                 |  s_jezyka_przedm
```

- napis wejściowy jest poprawny jeżeli da się go *zredukować*, (lub gdy *wygeneruje*) *symbol początkowy* gramatyki
- EBNF dodaje mechanizm wyrażeń regularnych do BNF

Gramatyka – przykład

```
<wyrazenie> := <liczba> |
               <liczba> '*' <liczba> |
               <liczba> '/' <liczba> |
               <liczba> '+' <liczba> |
               <liczba> '-' <liczba>
<liczba> := <cyfra>+
<cyfra> := '1'|'2'|'3'|'4'|'5'|
           '6'|'7'|'8'|'9'|'0'
```

Czym jest DTD

- DTD to *Document Type Definition/Declaration*
- w SGML definiuje w pełni strukturę dokumentu
- w XML są nałożone pewne ograniczenia
- przy jej pomocy można sprawdzać poprawność składniową dokumentu
- można w niej deklarować/definiować: encje, alementy, atrybuty
- DTD może być: załączone w dokumencie XML, dostępne w systemie, lub przez URL

Przykład DTD

```
<!DOCTYPE cds [
    <!ELEMENT cds (cd)+ >
    <!ELEMENT cd  (title, band, song*) >
    <!ELEMENT title (#PCDATA) >
    <!ELEMENT band  (#PCDATA) >
    <!ELEMENT song  (#PCDATA) >
    <!ATTLIST cd
        num CDATA #IMPLIED
    >
    <!ENTITY instr "instrumental version" >
]>
```

Odnoszenie się do DTD 1

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE cds [
    <!ELEMENT cds (cd)+ >
...

<?xml version="1.0" standalone="yes"?>
<!DOCTYPE cds SYSTEM "cds.dtd">
<?xml-stylesheet type="text/css"
                 href="cds2.css"?>
<cds>
...
```

Przykład DTD

```
<!-- elementy -->
<!ELEMENT cds (cd)+ >
<!ELEMENT cd  (title, band, song*) >
<!ELEMENT title (#PCDATA) >
<!ELEMENT band  (#PCDATA) >
<!ELEMENT song  (#PCDATA) >
<!-- atrybuty -->
<!ATTLIST cd
        num CDATA #IMPLIED
>
<!-- encje -->
<!ENTITY  instr "wykonanie instrumentalne" >
```

Dokument XML

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE cds SYSTEM "cds.dtd">
<?xml-stylesheet type="text/css" href="cds2.css"?>
<cds> <cd> <title>Cryptic Writings</title>
    <band>Megadeth</band>
    <song>Trust</song></cd>
  <cd num="2">
    <title>Aenima</title> <band>Tool</band>
    <song>Third Eye</song></cd>
  <cd>
    <title>Sabbath Bloody Sabbath</title>
    <band>Black Sabbath</band>
    <song>Fluff &instr;</song></cd>
</cds>
```

Składnia DTD

DTD może zawierać deklaracje:

```
<!ELEMENT nazwa (skladnia) >
<!ATTLIST element
        atr1 typ wartosc
        atr2 typ wartosc >
<!ENTITY nazwa tresc >
```

Składnia DTD

składnia elementu może mieć postać:

```
(el*, el+, el1? | el2)
(elA+ | elB?)*
(#PCDATA)
(el+ | #PCDATA)?
```

Składnia DTD

- typy atrybutów:
  `CDATA ID IDREF IDREFS NMTOKEN NMTOKENS ENTITY ETITIES`
- wartości atrybutów: `#REQUIRED, #IMPLIED, #FIXED`
- encje: wewnętrzne (napisy), zewnętrze (odwołania do innych plików `SYSTEM "plik"`, parametryczne

Łączenie różnych DTD

- tylko DTD wewnętrzne:

```
<!DOCTYPE cds [
    <!ELEMENT cds (cd)+ >
              ]>
```

- tylko DTD zewnętrzne:

```
<!DOCTYPE cds SYSTEM "cds.dtd">
```

- łączenie DTD:

```
<!DOCTYPE cds SYSTEM "cds.dtd" [
    <!ATTLIST song
        flavour CDATA #IMPLIED >
    <!ENTITY  concept "concept album" > ]>
```

DTD zewnętrzne

Systemowe:

```
<!DOCTYPE cds SYSTEM uri>
<!DOCTYPE cds SYSTEM "cds.dtd">
<!DOCTYPE cds SYSTEM
 "http://www.moje.org/cds.dtd">
```

Publiczne:

```
<!DOCTYPE nazwa PUBLIC
 "rodzaj//wlasciciel//opis//jezyk">
<!DOCTYPE html PUBLIC
 "-//W3C//DTD XHTML 1.0 Transitional//EN">
```

Spis Treści

XML Schema

- XML Schema mają zastąpić DTD
- DTD jest zapisywany w osobnymy języku (opartym o BNF)
- XML Schema są zapisywane w XML!
- mają większe możliwości rozszerzenia i rozbudowy
- mogą być same przetwarzane jak każdy inny dokument XML!

Przykład

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="cds2.css"?>
<cds
xmlns="http://home.agh.edu.pl/sth"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://home.agh.edu.pl/sth/
                    cds.xsd">
  <cd>
    <title>Cryptic Writings</title>
    <band>Megadeth</band>
    <song>Trust</song></cd>
  <cd num="2">
    <title>Aenima</title> <band>Tool</band>
    <song>Third Eye</song></cd>
</cds>
```

Przykład

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://home.agh.edu.pl/sth"
xmlns="http://home.agh.edu.pl/sth"
elementFormDefault="qualified">
<xs:attribute name="num" type="xs:string"/>
<xs:element name="cd">
  <xs:complexType> <xs:sequence>
      <xs:element name="title" type="xs:string"/>
      <xs:element name="band" type="xs:string"/>
      <xs:element name="song" type="xs:string"/>
    </xs:sequence>
    <xs:attribute ref="num" use="optional"/>
  </xs:complexType> </xs:element>
<xs:element name="cds">
    <xs:complexType> <xs:sequence>
      <xs:element ref="cd" maxOccurs="unbounded"/>
    </xs:complexType>
```

Spis Treści

Style

- CSS,
- XSL – eXtensible Stylesheet Language – transformacja do HTML,
  - XSLT – eXtensible Stylesheet Language Transformations.

CSS

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?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>
</CATALOG>
```

CSS

```
CATALOG {
  background-color: #ffffff;
  width: 100%;}
CD {
  display: block;
  margin-bottom: 30pt;
  margin-left: 0;}
TITLE {
  color: #FF0000;
  font-size: 20pt;}
ARTIST {
  color: #0000FF;
  font-size: 20pt; }
COUNTRY,PRICE,YEAR,COMPANY {
  display: block;
  color: #000000;
  margin-left: 20pt;}
```

CSS, selektory dla XML

```
E[atrybut]                element E, z atrybutem  atrybut
E[atrybut="wartosc"]      element E, z atrybutem atrybut
                          o wartości wartosc
E[atrybut~="wartosc"]     element E, z atrybutem atrybut
                          o wartości będącej listą słów
                          zawierających wartosc
                          (oddzielonych białymi znakami)
```

Spis Treści

Czym jest XHTML

- *eXtensible HyperText Markup Language*
- ma w pełni zastąpić HTML
- składnia identyczna do HTML 4.01
- dokładniejsza, bardziej restrykcyjna specyfikacja
- XHTML to aplikacja XML (zdefiniowany w XML)
- pełne możliwości walidacji
- style w CSS

Różnice wzgl. HTML

- wszystkie el. *muszą* być poprawnie zagnieżdżone
- dokumenty muszą być *dobrze sformułowane*
- nazwy znaczników pisane *małymi literami*
- wszystkie znaczniki i atrybuty muszą być *zamknięte*
- wartości atrybutów muszą być *cytowane*
- w prologu musi być *odniesienie do DTD*

HTML: id vs. name

- Wszystkie `id` oraz `name` dzielą wspólną przestrzeń nazw.
- `id` jest więcej niż *nazwą* dla hiperpołączeń, umożliwia stosowanie stylów/programowania,
- niektóre (starsze) przeglądarki mogą mieć kłopoty z odnośnikami do `id`,
- `name` pozwala na stosowanie dowolnych znaków, w przypadku `id` jedynie litery/cyfry.
- Dla XHTML 1.0 `name` nie jest zalecane do identyfikacji elementów.
- `name` może nie być unikalne, różna semantyka dla różnych elementów.

XHTML

- XML ma zastąpić HTML
- potrzeba płynnego przejścia
- XHTML – HTML 4 zdefiniowany w XML 1.0
- jedynie drobne różnice składniowe i semantyczne
- pomijalne, z p. widzenia bieżących narzędzi
- XML jest podstawą współczesnych technologii internetowych

Źródła

- www.w3c.org
- www.xml.org
- www.xml.com
- xml.coverpages.org
- xml.oreilly.com
- www.ibm.com/developerworks/xml
- www.w3schools.com