

Prezentacja dokumentów XML

Patryk Czarnik

Instytut Informatyki UW

XML i nowoczesne technologie zarządzania treścią – 2011/12

Arkusze stylu

- Rozdzielenie treści i wyglądu
- Przypisanie stylu do dokumentu

CSS

- Idea i zastosowania
- Struktura arkusza
- Formatowanie

XSL

- Wizualizacja dzięki przekształceniu
- XSLT – wprowadzenie
- XSL-FO

HTML

- Idea i historia
- XHTML

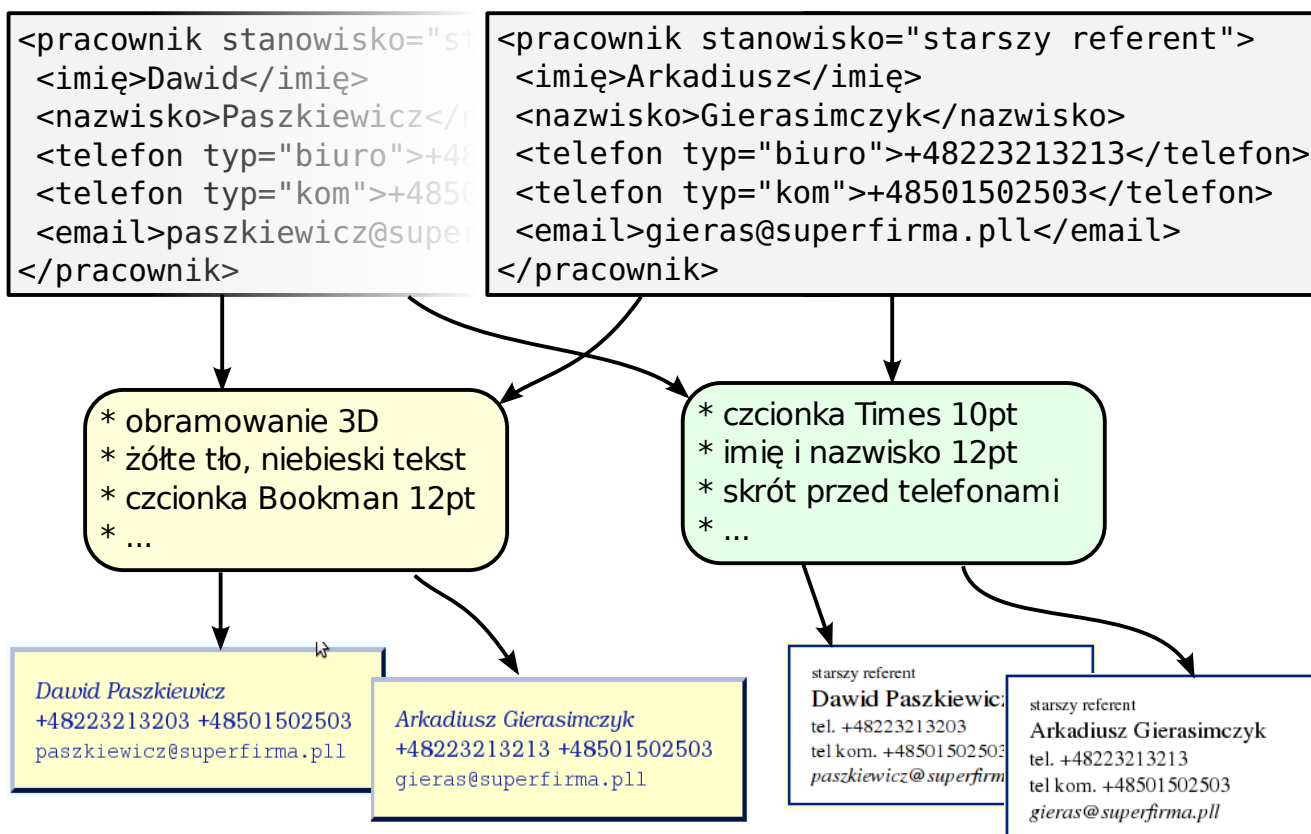
Oddzielenie treści od wyglądu

- ▶ Zgodnie z ideą XML w dokumentach tylko:
 - ▶ treść/dane,
 - ▶ znaczniki dla struktury, znaczenia („znakowanie semantyczne”).
- ▶ Brak informacji o wyglądzie.

Jak prezentować?

- ▶ Interpretacja znanych typów dokumentów, np.:
 - ▶ aplikacja okienkowa,
 - ▶ aplikacja korzystająca z API do tworzenia PDF.
- ▶ Zewnętrzne arkusze stylu:
 - ▶ wspólne ramy odpowiednie dla dowolnych typów dokumentów.

Idea arkusza stylu



Zalety oddzielenia treści od wyglądu

- ▶ Łatwiejsza analiza danych źródłowych i inne już znane zalety znakowania semantycznego.
- ▶ Możliwość ponownej prezentacji:
 - ▶ po zmianie danych,
 - ▶ dla innego dokumentu o takiej samej strukturze.
- ▶ Zmiana wyglądu dokonywana poza dokumentem źródłowym, raz dla całej klasy dokumentów.
- ▶ Możliwość zdefiniowania wielu arkuszy stylu dla danej klasy dokumentów w zależności od:
 - ▶ przeznaczenia,
 - ▶ medium (ekran, wydruk, dźwięk),
 - ▶ szczegółowości (pełen raport, podsumowanie itp.),
 - ▶ preferencji czytelnika (rozmiar czcionki, kolory...).

Standardy związane z prezentacją XML

- ▶ Przypisanie arkusza stylu do dokumentu:
 - ▶ *Associating Style Sheets with XML documents*
- ▶ Języki do zapisywania arkuszy stylu:
 - ▶ DSSSL (historyczny, stosowany dla SGML)
Document Style Semantics and Specification Language
 - ▶ CSS
Cascading Style Sheets
 - ▶ XSL
Extensible Stylesheet Language

Przypisanie stylu do dokumentu

- ▶ Za pomocą instrukcji przetwarzania `xml-stylesheet`.
- ▶ Opisane w rekomendacji W3C *Associating Style Sheets with XML documents*.

Jeden styl

```
<?xml-stylesheet href="normalny.xsl" type="text/xsl"?>
```

Style alternatywne

```
<?xml-stylesheet title="Niebieski"
  type="text/css" href="niebiesko.css" ?>
```

```
<?xml-stylesheet title="Żółty" alternate="yes"
  type="text/css" href="zolto.css" ?>
```

```
<?xml-stylesheet title="Malutki" alternate="yes"
  type="text/css" href="compact.css" ?>
```

Cascading Style Sheets

- ▶ Początki idei arkuszy stylu: lata 70-te XX wieku.
- ▶ Początki CSS: 1994.
- ▶ Rekomendacja CSS Level 1: grudzień 1996.
- ▶ Rekomendacja CSS Level 2: maj 1998
 - ▶ CSS jaki znamy (ale nie pewna przeglądarka).
- ▶ CSS Level 3 – ciągle rozwijane:
 - ▶ modularyzacja (niektóre moduły już zatwierdzone),
 - ▶ nowe możliwości.
- ▶ CSS 2.1 – rekomendacja z lipca 2011:
 - ▶ uszczegółowienie i poprawki specyfikacji CSS 2,
 - ▶ dostosowanie specyfikacji do powszechnie stosowanych praktyk,
 - ▶ rezygnacja z rzeczy, które się nie sprawdziły.

Zastosowania CSS

- ▶ Pierwsze i główne zastosowanie: styl dla stron WWW.
- ▶ Oddzielenie treści od formatowania dla HTML.
- ▶ (Proste) arkusze stylu dla XML.
- ▶ Od CSS 2 bardzo ważne „ideologicznie”:
 - ▶ wsparcie dla alternatywnych metod prezentacji treści (np. czytanie na głos),
 - ▶ umożliwienie **czytelnikom** podania własnego stylu (np. większej czcionki i innych kolorów dla słabo widzących).

Przykładowy dokument XML (fragment)

```
<oddział id="ksi">
  <nazwa>Księgowość</nazwa>

  <pracownik stanowisko="starszy referent" id="102103">
    <imię>Dawid</imię><nazwisko>Paszkiewicz</nazwisko>
    <telefon typ="biuro">+48223213203</telefon>
    <telefon typ="kom">+48501502503</telefon>
    <email>paszkiewicz@superfirma.pl</email>
  </pracownik>

  <pracownik stanowisko="kierownik" id="102104">
    <imię>Monika</imię><nazwisko>Domżałowicz</nazwisko>
    <telefon typ="biuro">+48223213200</telefon>
    <telefon typ="kom">+48501502513</telefon>
    <email>mdom@superfirma.pl</email>
  </pracownik>
  ...
</oddział>
```

Przykład arkusza

```
pracownik {
  display: block;
  margin: 10px auto 10px 30px;
  padding: 0.75em 1em;
  width: 200px;
  border-style: solid;
  border-width: 2px;
  border-color: #002288;
  background-color: #FFFFFF; }

pracownik[stanowisko='kierownik'] {
  background-color: #DDFFDD; }

imię, nazwisko {
  display: inline;
  font-size: larger; }

pracownik[stanowisko='kierownik'] nazwisko {
  font-weight: bold; }
```

Wynik formatowania CSS

Księgowość

starszy referent
Dawid Paszkiewicz
tel. +48223213203
tel kom. +48501 502503
paszkiewicz@superfirma.pll

kierownik
Monika Domżałowicz
tel. +48223213200
tel kom. +48501 502513
mdom@superfirma.pll

Selektory (przykłady)

- ▶ imię – nazwa elementu,
- ▶ imię, nazwisko – oba elementy,
- ▶ oddział nazwisko – potomek,
- ▶ oddział > nazwa – dziecko,
- ▶ A + B – następnik,
- ▶ imię:first-child – pierwsze dziecko,
- ▶ pracownik[stanowisko] – test istnienia atrybutu,
- ▶ pracownik[stanowisko='kierownik'] – test wartości atrybutu,
- ▶ pracownik[funkcje~='kierownik'] – wartość występuje na liście,
- ▶ pracownik#k12 – wartość atrybutu zadeklarowanego jako ID,
- ▶ ol.pracownicy – równoważne ol[class~='pracownicy'] (tylko HTML).

Styl zależny od rodzaju medium

Przykład

```
@media print {
  pracownik {
    background-color: white;
    font-family: serif;
  }
}

@media screen {
  pracownik {
    background-color: yellow;
    font-family: sans-serif;
  }
}

@media all {
  pracownik {
    border-style: solid;
  }
}
```

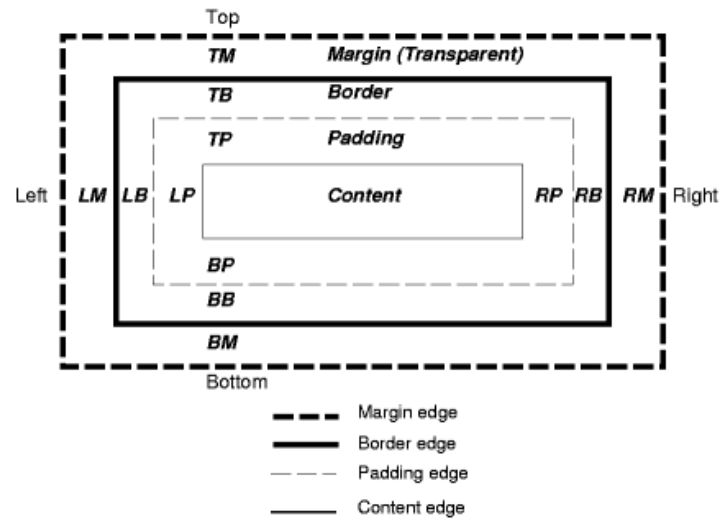
Własność `display`

- ▶ Rodzaj obiektu wizualnego reprezentującego element.
- ▶ Możliwe wartości: `inline`, `block`, `list-item`, `run-in`, `inline-block`, `table`, `table-cell`, ..., `none`,
- ▶ Podstawowa własność w przypadku wizualizacji XML.
- ▶ Rzadko stosowane dla HTML.

Pudełka i wyrównanie

- ▶ Zagnieżdżenie bloków odpowiada zagnieżdżeniu elementów w dokumencie.
- ▶ Możliwe ręczne pozycjonowanie.
- ▶ `margin`, `padding` – margines zewnętrzny i wewnętrzny,
- ▶ `border-style`, `border-color`, `border-width` – obramowanie,
- ▶ `position` (`static`, `relative`, `absolute`, `fixed`) – sposób pozycjonowania,
- ▶ `left`, `right`, `top`, `bottom` – pozycja,
- ▶ `width`, `height`, `min-width`, `max-height`, ... – rozmiar.

Pudełka



Źródło: W3C, Rekomendacja CSS Level 2

Przykład z marginesami i obramowaniem

```
pracownik {  
  display: block;  
  margin: 10px auto 10px 30px;  
  padding: 0.75em 1em;  
  width: 200px;  
  border-style: solid;  
  border-width: 2px;  
  border-color: #002288;  
  background-color: #FFFFFF;  
}
```

Tekst i czcionka

- ▶ `color`, `background-color`, `background-image` – kolor i tło,
- ▶ `font-family` – nazwa czcionki oraz `serif`, `sans-serif`, `monospace` ...
- ▶ `font-size`,
- ▶ `font-style`, `font-weight`,
- ▶ `text-decoration`,
- ▶ `text-align`.

Przykład z czcionkami i kolorami

```
firma {
  display: block;
  background-color: #EEEEFF;
  color: rgb(0, 0, 33%);
  font-family: 'Bookman', serif;
  font-size: 14pt;
}

firma > nazwa {
  font-size: 1.5em;
  font-family: 'Verdana', 'Arial', sans-serif;
  font-weight: bold;
  text-align: center;
}

oddział > nazwa {
  font-size: 1.3em;
  font-family: 'Verdana', 'Arial', sans-serif;
  font-weight: bold;
  font-style: italic;
}
```

Zawartość generowana

- ▶ Wstawianie tekstów nie będących zawartością tekstową dokumentu.
- ▶ Dostęp do wartości atrybutów.
- ▶ Automatyczne numeracje.

Przykład

```
pracownik:before {
  content: attr(stanowisko);
}

telefon[typ='biuro']:before {
  content: 'tel. ';
}

telefon[typ='kom']:before {
  content: 'tel kom. ';
}
```

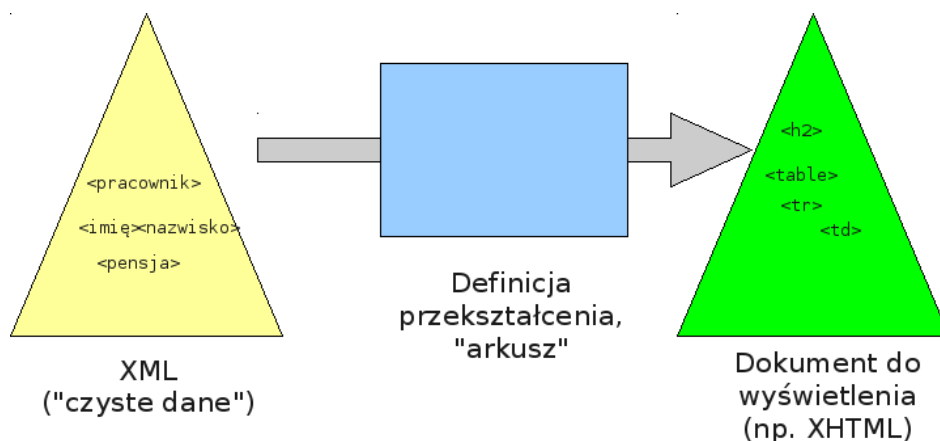
Możliwości i zalety CSS

- ▶ Ogromne możliwości wpływania na wygląd.
- ▶ Rozróżnianie elementów ze względu na:
 - ▶ nazwę,
 - ▶ położenie w drzewie dokumentu,
 - ▶ występowanie atrybutów,
 - ▶ wartości atrybutów.
- ▶ Szerokie wsparcie:
 - ▶ przeglądarki internetowe,
 - ▶ narzędzia do tworzenia arkuszy.
- ▶ Łatwo pisać proste arkusze :)
(inaczej mówiąc – niska bariera technologiczna dla nowicjuszy)

Ograniczenia CSS

- ▶ Tylko wizualizacja danych
 - ▶ a nie np. przedstawienie w innym formacie.
- ▶ Niemożliwe (w CSS Level 2):
 - ▶ powtórzenie tego samego fragmentu kilkakrotnie,
 - ▶ rozróżnianie elementów ze względu na ich zawartość,
 - ▶ zaawansowane warunki logiczne,
 - ▶ przetwarzanie danych (np. zsumowanie wartości liczbowych),
 - ▶ dostęp do wielu dokumentów na raz.
- ▶ Trudne lub nienaturalne w CSS:
 - ▶ wyświetlenie wartości atrybutów,
 - ▶ wypisanie elementów w innej kolejności niż w dokumencie,
 - ▶ wyróżnienie elementów nie spełniających pewnego warunku.

Wizualizacja dzięki przekształceniu

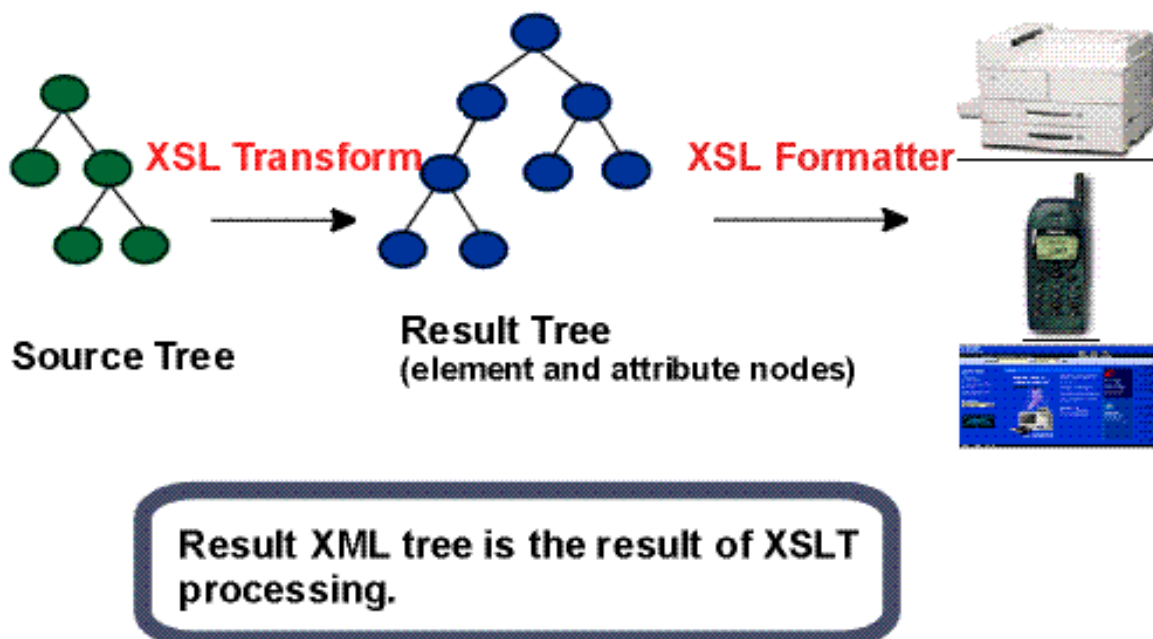


- ▶ Sposobem prezentacji dokumentu może być przekształcenie do takiego formatu XML, który potrafimy prezentować.
- ▶ Prezentowalne formaty XML:
 - ▶ XHTML,
 - ▶ XSL Formatting Objects.

Extensible Stylesheet Language (XSL)

- ▶ Zdefiniowany w rekomendacjach (wersje 1.0 w 1999 i 2001):
 - ▶ **XSL** (ogólne ramy, język XSL Formatting Objects),
 - ▶ **XSLT** (arkusze – przekształcenia dokumentów XML),
 - ▶ **XPath** (język wyrażeń zawierający ścieżki do adresowania fragmentów dokumentu).
- ▶ Arkusz stylu mówi jak przekształcać dany typ dokumentu do dokumentu XSL-FO.
- ▶ W praktyce przekształcenia także do innych formatów, często (X)HTML.
- ▶ Zazwyczaj dany arkusz jest odpowiedni dla dokumentów XML określonego rodzaju (konkretnego zastosowania XML).

Idea XSL



XSLT – status

- ▶ Powstał w ramach standardu XSL.
- ▶ Zastosowania wykraczają poza wizualizację XML.
- ▶ Wersja 1.0:
 - ▶ listopad 1999, powiązane z XPath 1.0,
 - ▶ szerokie wsparcie w oprogramowaniu.
- ▶ Wersja 2.0:
 - ▶ styczeń 2007, powiązane z XPath 2.0 i XQuery 1.0,
 - ▶ głębsze podstawy teoretyczne, większe możliwości,
 - ▶ mniejsze (ale istniejące) wsparcie.

XSLT – dostępność

- ▶ Procesory XSLT 2.0:
 - ▶ Saxon
 - ▶ biblioteki dla Javy i .NET, aplikacje command-line,
 - ▶ darmowa (Open Source) wersja podstawowa,
 - ▶ komercyjna wersja *schema aware*.
 - ▶ XML Spy (komercyjny program okienkowy).
- ▶ Procesory XSLT 1.0:
 - ▶ przeglądarki internetowe (co najmniej IE, Mozilla/Firefox, Opera),
 - ▶ Xalan (biblioteki dla Javy i C++),
 - ▶ xsltproc, część pakietu libxml (w C, zasadniczo dla Linuxa),
 - ▶ XML-owe rozszerzenia silników baz danych,
 - ▶ ...
- ▶ Narzędzia do tworzenia arkuszy:
 - ▶ zwykły edytor tekstu – mnóstwo, w tym wiele darmowych,
 - ▶ środowiska programistyczne, w tym Eclipse,
 - ▶ komercyjne narzędzia do XML-a (XML Spy, oXygen, ...).

XSLT – przykład arkusza

1/2

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" encoding="utf-8" />

  <xsl:template match="/">
    <html>
      <head>
        <title>Lista pracowników</title>
      </head>
      <body>
        <h1>Lista pracowników</h1>
        <ul>
          <xsl:apply-templates
            select="//pracownik"/>
        </ul>
      </body>
    </html>
  </xsl:template>
  ...
```

XSLT – przykład arkusza

2/2

```
...
  <xsl:template match="pracownik">
    <li>
      <xsl:value-of select="imię"/>
      <xsl:value-of select="nazwisko"/>
      (<xsl:value-of select="telefon[typ='kom']"/>)
    </li>
  </xsl:template>
</xsl:stylesheet>
```

XSLT – struktura arkusza

- ▶ **Arkusz** (*stylesheet*) składa się z szablonów.
- ▶ **Szablon** (*template*) mówi jak przekształcać węzeł dokumentu wejściowego na fragment dokumentu wynikowego.
- ▶ Wewnątrz szablonu:
 - ▶ tekst i elementy spoza przestrzeni nazw XSLT → przepisywane do wyniku,
 - ▶ **instrukcje XSLT** → sterowanie przetwarzaniem, dodatkowe operacje,
 - ▶ **ścieżki XPath** w niektórych instrukcjach → dostęp do dokumentu źródłowego, sprawdzanie warunków, arytmetyka itp.
- ▶ XSLT jest tak naprawdę językiem programowania stworzonym z myślą o przekształcaniu dokumentów XML.

XSLT – idea działania

- ▶ Przekształcenie na poziomie drzewa dokumentu.
- ▶ Jako pierwszy uruchamiany szablon dla korzenia dokumentu
 - ▶ taki szablon istnieje, nawet gdy sami go nie napiszemy.
- ▶ Instrukcje `apply-templates` wewnątrz szablonu powodują przejście po drzewie dokumentu źródłowego (zwykle w głąb dokumentu) i uruchamianie szablonów dla kolejnych węzłów.
- ▶ Szablony dopasowywane do węzłów w zależności od rodzaju węzła, nazwy elementu, położenia w drzewie dokumentu i innych warunków.
- ▶ Możliwe przepisywanie treści ze źródła, tworzenie nowych węzłów i treści, sprawdzanie warunków, analiza danych itd.

Rola XPath w XSLT

- ▶ Dostęp do dokumentu źródłowego możliwy dzięki **ścieżkom XPath**.
- ▶ Możliwe:
 - ▶ umieszczenie fragmentu dokumentu lub wartości odczytanej z dokumentu,
 - ▶ umieszczenie wartości „obliczonej” wyrażeniem XPath,
 - ▶ sprawdzenie warunku logicznego.
- ▶ Dostępne w XPath:
 - ▶ ścieżki (dostęp do węzłów dokumentu),
 - ▶ operatory logiczne i arytmetyczne,
 - ▶ funkcje na liczbach, napisach i inne,
 - ▶ więcej w wersji 2.0.
- ▶ XSLT 1.0 używa XPath w wersji 1.0.
- ▶ XSLT 2.0 używa XPath w wersji 2.0.

XPath w arkuszu XSLT – przykład

```
...  
<xsl:template match="pracownik">  
  <li>  
    <xsl:value-of select="imię"/>  
    <xsl:value-of select="./nazwisko"/>  
    (<xsl:value-of select="telefon[typ='kom']"/>)  
  </li>  
</xsl:template>  
</xsl:stylesheet>
```

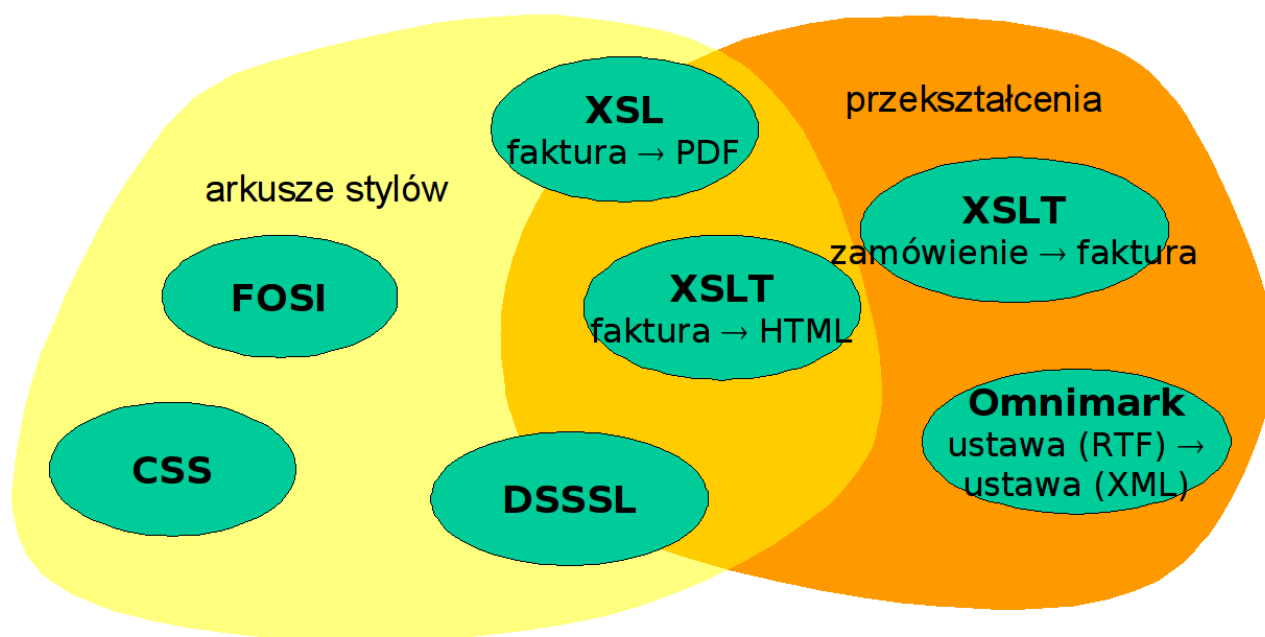
Ścieżki – typowe zastosowanie XPath

- ▶ `/firma/oddział/pracownik`
- ▶ `//pracownik`
- ▶ `/firma/oddział[nazwa = 'Księgowość']`
- ▶ `/firma/oddział[@id = 'ksi']/pracownik[3]`
- ▶ `./nazwisko`
- ▶ `nazwisko`
- ▶ `../pracownik[stanowisko = 'kierownik']/nazwisko`

XSLT – wynik przekształcenia

- ▶ XSL Formatting Objects:
 - ▶ zgodnie z ideą standardu XSL,
 - ▶ przydatne np. dla wydruków.
- ▶ HTML i XHTML:
 - ▶ najbardziej popularne,
 - ▶ wygodne w przypadku publikacji przez WWW / wyświetlania na ekranie.
- ▶ Dowolny XML, np.:
 - ▶ tłumaczenie do innego / nowego formatu,
 - ▶ wydobywanie i przetwarzanie danych (alternatywa dla XQuery),
 - ▶ XSLT w wyniku przekształcenia XSLT.
- ▶ Zwykły tekst, np.:
 - ▶ CSV i inne tekstowe formaty danych,
 - ▶ skrypty i pliki konfiguracyjne na podstawie dokumentów XML,
 - ▶ tłumaczenie dokumentów do innego formatu tekstowego (LaTeX?).

Przekształcenia a style



Źródło: Szymon Ziolo, XML i nowoczesne technologie zarządzania treścią

XSL Formatting Objects

- ▶ Zastosowanie XML służące do prezentacji.
- ▶ Zorientowany na wydruk:
 - ▶ szablony stron,
 - ▶ automatyczny podział na strony.
- ▶ Dość przegadane... (np. listy).

Uwaga

Normalnie nie pisze się dokumentów w XSL-FO, tylko przekształcenia do XSL-FO.

Struktura dokumentu XSL-FO

```
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

<fo:layout-master-set>
  <fo:simple-page-master master-name="moja-strona">
    <fo:region-body />
  </fo:simple-page-master>
</fo:layout-master-set>

<fo:page-sequence master-reference="moja-strona">
  <fo:flow flow-name="xsl-region-body">
    <fo:block>Hello World!</fo:block>
  </fo:flow>
</fo:page-sequence>

</fo:root>
```

Przykład przekształcenia do XSL-FO (fragment 1)

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <xsl:output method="xml" encoding="utf-8"/>

  <xsl:template match="/">
    <fo:root>

      <fo:layout-master-set>
        <fo:simple-page-master master-name="A4" ...>
          ...
        </fo:simple-page-master>
      </fo:layout-master-set>

      <fo:page-sequence master-reference="A4">
        <fo:flow flow-name="xsl-region-body">
          <xsl:apply-templates />
        </fo:flow>
      </fo:page-sequence>
    </fo:root>
  </xsl:template>
  ...
</xsl:stylesheet>
```

Przykład przekształcenia do XSL-FO (fragment 2)

```
<xsl:template match="pracownik">
  <fo:block
    border-width="1.5pt"
    border-style="solid"
    border-color="#664400"
    background-color="#FFFFFFEE">
    ...
    <fo:block>
      <xsl:apply-templates select="telefon"/>
    </fo:block>
  </fo:block>
</xsl:template>

<xsl:template match="telefon">
  <fo:block>
    <xsl:choose>
      <xsl:when test="@typ='kom'">kom. </xsl:when>
      <xsl:otherwise>tel. </xsl:otherwise>
    </xsl:choose>
    <xsl:apply-templates />
  </fo:block>
</xsl:template>
```

Wizualizacja przykładu

Księgowość

Dawid Paszkiewicz
tel. +48223213203
kom. +48501502503
paszkiewicz@superfirma.pl

Monika **Domżałowicz**
tel. +48223213200
kom. +48501502513
m.dom@superfirma.pl

Hierarchia „pudełek”

- ▶ strona
- ▶ obszar (*region*) – pięć predefiniowanych:
 - ▶ *region-body* – główna zawartość strony,
 - ▶ *region-before* – nad,
 - ▶ *region-after* – pod,
 - ▶ *region-start* – po lewej,
 - ▶ *region-end* – po prawej.
- ▶ blok
- ▶ wiersz (*line*)
- ▶ wewnątrz wiersza (*inline*)

Szablon strony (*page-master*)

- ▶ Określają układ pojedynczej strony.
- ▶ Dokument może zostać podzielony na wiele takich stron (gdy zawartość się nie mieści)

Przykład

```
<fo:simple-page-master master-name="A4"
  page-width="297mm" page-height="210mm"
  margin-top="1cm"   margin-bottom="1cm"
  margin-left="1cm"  margin-right="1cm">
```

```
  <fo:region-body    margin="3cm"/>
  <fo:region-before  extent="2cm"/>
  <fo:region-after   extent="2cm"/>
  <fo:region-start   extent="2cm"/>
  <fo:region-end     extent="2cm"/>
```

```
</fo:simple-page-master>
```

Zawartość stron

- ▶ `page-sequence` – pewna liczba stron dokumentu.
- ▶ Zawartość elementu `flow` przepływa na kolejne strony.
- ▶ Zawartość elementu `static-content` powtarza się na stronach.
- ▶ Atrybut `flow-name` – do którego obszaru strony ma trafić zawartość.

Fragment arkusza

```
<fo:page-sequence master-reference="A4">
  <fo:static-content flow-name="xsl-region-before">
    Pracownicy firmy <xsl:value-of select="nazwa" />
  </fo:static-content>

  <fo:flow flow-name="xsl-region-body">
    <xsl:apply-templates />
  </fo:flow>
</fo:page-sequence>
```

Wygląd fragmentów dokumentu

- ▶ Wygląd i własności bloków (różnych poziomów) określone w atrybutach:
 - ▶ `margin`, `padding`, `border-style` ...
 - ▶ `background-color`, `background-image` ...
 - ▶ `font-family`, `font-weight`, `font-style`, `font-size` ...
 - ▶ `text-align`, `text-align-last`, `text-indent`, `start-indent`, `end-indent`, `wrap-option`, `break-before` ...
- ▶ Bardzo podobne do własności CSS.

Listy – przykład

```
<fo:list-block>
  <fo:list-item>
    <fo:list-item-label>
      <fo:block>Imię: </fo:block>
    </fo:list-item-label>
    <fo:list-item-body>
      <fo:block margin-left="15em">Dawid</fo:block>
    </fo:list-item-body>
  </fo:list-item>
  <fo:list-item>
    <fo:list-item-label>
      <fo:block>Nazwisko: </fo:block>
    </fo:list-item-label>
    <fo:list-item-body>
      <fo:block margin-left="15em">Paszkiewicz</fo:block>
    </fo:list-item-body>
  </fo:list-item>
</fo:list-block>
```

Wizualizacja przykładu – wersja z listami

Księgowość

- * Stanowisko: starszy referent
Imię: Dawid
Nazwisko: Paszkiewicz
tel.: +48223213203
kom.: +48501502503
Email: *paszkiewicz@superfirma.pl*

- * Stanowisko: kierownik
Imię: Monika
Nazwisko: Domżałowicz
tel.: +48223213200
kom.: +48501502513
Email: *m.dom@superfirma.pl*

Tabele – przykład

```
<fo:table border="solid 2pt black">
  <fo:table-header>
    <fo:table-row>
      <fo:table-cell><fo:block font-weight="bold">Nazwisko
        </fo:block></fo:table-cell>
      <fo:table-cell><fo:block font-weight="bold">Imię
        </fo:block></fo:table-cell>
      ...
    </fo:table-row>
  </fo:table-header>
  <fo:table-body>
    <fo:table-row>
      <fo:table-cell><fo:block>Paszkiewicz</fo:block></fo:table-cell>
      <fo:table-cell><fo:block>Dawid</fo:block></fo:table-cell>
      ...
    </fo:table-row>
  ...
</fo:table>
```

Wizualizacja przykładu – wersją z tabelą

Księgowość

Nazwisko	Imię	Telefony	Email
Paszkiewicz	Dawid	+48223213203 +48501502503	<i>paszkiewicz@superfirma</i>
Domżałowicz	Monika	+48223213200 +48501502513	<i>mdom@superfirma.pl</i>
Kącki	Marek	+48223213212 +48501502524	<i>kacki@superfirma.pl</i>
Woźniak	Tomasz	+48223213234 +48501502544	<i>wozniak@superfirma.pl</i>
Chmielnicki	Maciej	+48223213236 +48501502527	<i>chmiel@superfirma.pl</i>
Ślusarek	Patrycja	+48223213216 +48501502594	<i>paszkiewicz@superfirma</i>
Koper	Robert	+48501502111	<i>paszkiewicz@superfirma</i>

XSL-FO – dostępność

- ▶ Oprogramowanie komercyjne:
 - ▶ Antenna House XSL Formatter
 - ▶ RenderX
 - ▶ Ecrion
 - ▶ Lunasil LTD Xinc
 - ▶ ...
- ▶ Oprogramowanie otwarte:
 - ▶ Apache FOP
 - ▶ xmlroff
 - ▶ ?

XSL-FO – krytyka

- ▶ Główne zalety XSL-FO:
 - ▶ naturalny w przypadku transformacji XSLT,
 - ▶ najbardziej bezpośredni sposób do uzyskania wydruku (pliku PDF itp.) z danych zapisanych w XML,
 - ▶ ogólne zalety arkusza stylu w stosunku do narzędzi formatujących „na sztywno”
- ▶ Główne wady XSL-FO:
 - ▶ zbyt skomplikowany do prostych zastosowań,
 - ▶ zbyt ograniczony do złożonych zastosowań, m.in.:
 - ▶ brak informacji zwrotnej o rozłożeniu na strony i wykorzystania tego w formatowaniu
 - ▶ dostępne tylko wymienione w standardzie aspekty formatowania (a często chcemy ręcznie zrobić „coś jeszcze”)

Historia HTML

- ▶ Początki HTML:
 - ▶ Tim Berners-Lee, CERN, początek lat 1980-tych,
 - ▶ ENQUIRE – język dla dokumentów technicznych.
- ▶ Od 1989 – zastosowanie HTML w Internecie.
- ▶ 1993 – HTML 2.0 zdefiniowany jako zastosowanie SGML.
- ▶ HTML 3.2 – 1997:
 - ▶ dużo znaczników formatujących bezpośrednio w dokumentach.
- ▶ HTML 4 – 1999:
 - ▶ ograniczony zestaw znaczników,
 - ▶ nacisk na strukturę i semantykę,
 - ▶ CSS zamiast znaczników i atrybutów formatujących.

Znaczniki HTML

- ▶ Strukturalne:
 - ▶ h1 – h6, p,
 - ▶ ol, ul, li, dl, dt, dd,
 - ▶ table, tr, td, th ...
- ▶ Semantyczne:
 - ▶ q, strong, dfn, code ...
- ▶ Formatujące:
 - ▶ b, i, font, center

Znakowanie semantyczne w HTML

- ▶ Także w HTML należy stosować znakowanie semantyczne!
- ▶ Właściwe użycie nagłówków i akapitów.
- ▶ Stosowanie znaczników semantycznych jak `<q>`, ``.
- ▶ W przypadku braku odpowiedniego znacznika (dopiero!)
`<div class="cośtam">` lub ``

XHTML – główne idee

- ▶ XML zamiast SGML (łatwiejsze parsowanie, mniej wygodne pisanie).
- ▶ Modularyzacja (podział znaczników na grupy, stosowane niekoniecznie wszystkie na raz).
- ▶ Dzięki przestrzeniom nazw:
 - ▶ wykorzystanie istniejących standardów (XForms, SVG, MathML, ...),
 - ▶ dodawanie własnych znaczników.

Status HTML

- ▶ Obecnie zalecane:
 - ▶ HTML 4.01
 - ▶ XHTML 1.1
- ▶ Obecnie opracowywane:
 - ▶ HTML 5 – nowe zastosowania (video, canvas), wzbogacenie struktury (article, aside), ...
 - ▶ XHTML 2.0 – solidna przebudowa standardu, część HTML-a wyrzucona na korzyść zewnętrznych standardów (np. XForms). Obecnie projekt zawieszony, aż ukształtuje się HTML 5.

Typowy arkusz tworzący HTML

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html" encoding="utf-8" />
  <xsl:template match="/">
    <html>
      <head>
        <title><!-- JAKIŚ TYTUŁ --></title>
        <style type="text/css">
          <!-- TU OPISUJEMY STYL WYNIKOWEGO HTML-a -->
        </style>
      </head>
      <body>
        <xsl:apply-templates />
      </body>
    </html>
  </xsl:template>
  <!-- POZOSTAŁE SZABLONY -->
</xsl:stylesheet>
```

Typowy arkusz tworzący HTML – proste szablony

```
<xsl:template match="tytuł">
  <h2>
    <xsl:apply-templates />
  </h2>
</xsl:template>
```

```
<xsl:template match="osoba">
  <div class="osoba">
    <xsl:apply-templates />
  </div>
</xsl:template>
```

```
<xsl:template match="ważne">
  <strong>
    <xsl:apply-templates />
  </strong>
</xsl:template>
```