



# XML



# XML



- ▶ XML: uvod in osnovna sintaksa
- ▶ standardizacija oblike dokumenta XML
  - definicija tipa dokumenta (DTD)
  - sheme XML
- ▶ prikaz dokumentov XML
  - prikaz s slogovnimi predlogami CSS
  - prikaz s transformacijami XSLT
  - prikaz z branjem XML DOM
- ▶ razčlenjevalniki XML

# Razvoj XML

- ▶ HTML razvit 1990 z uporabo meta-označevalnega jezika Standard Generalized Markup Language (SGML). Problemi:
  - definirane samo osnovne značke, potreben večji besednjak značk glede na potrebe posameznih aplikacij
  - namen HTML je opis postavitve informacij v dokumentu
  - HTML ne opisuje POMENA podatkov
  - malo omejitev glede vrstnega reda (gnezdenje) značk
- ▶ možne rešitve:
  - določi svojo množico značk in razvij svoj označevalni jezik z uporabo SGML (kompleksno!)
  - razvij preprostejši meta-označevalni jezik za vsakdanjo uporabo →
    - XML je poenostavljena verzija SGML!
    - XML ni zamenjava za HTML, imata različne cilje (opis postavitve /meta-označevalni jezik za definiranje označevalnih jezikov)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ARTICLES SYSTEM
"D:\Project\clients\XML\Contents\Templarticlelist.dtd">
<?xml-stylesheet type="text/xsl"
href="D:\xmltohtml.xsl" ?>
<ARTICLES>
  <ARTICLE>
    <ARTICLEDATA>
      <TITLE>XML Demystified</TITLE>
      <AUTHOR>Jaidew</AUTHOR>
    </ARTICLEDATA>
  </ARTICLE>
  <ARTICLE>
    <ARTICLEDATA>
      <TITLE>XSLT Demystified</TITLE>
      <AUTHOR>X S Cel Tea</AUTHOR>
    </ARTICLEDATA>
  </ARTICLE>
  <ARTICLE>
    <ARTICLEDATA>
      <TITLE>C# Demystified</TITLE>
      <AUTHOR>Aleksy N</AUTHOR>
    </ARTICLEDATA>
  </ARTICLE>
</ARTICLES>
```

# XML

- ▶ XML se uporablja za definiranje označevalnih jezikov (primer: XHTML):

```
<?xml version = "1.0" encoding = "utf-8" ?>  
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

- ▶ XML specificira pravila za definiranje označevalnih jezikov, ne vsebuje značk, to morajo definirati uporabniki glede na namen uporabe
- ▶ XML določa podatkom **POMEN**, ne definira predstavitev
- ▶ zagotavlja univerzalen način za shranjevanje (in distribucijo) tekstovnih podatkov
- ▶ **razčlenjevalnik XML** je program, ki na vhodu vzame dokument XML, na izhodu pa poda njegove posamezne dele uporabniški aplikaciji
- ▶ **aplikacija XML** (strogo gledano) je nabor definiranih značk oz. konkretni jezik, definiran z XML

```
<?xml>  
.....  
Syntax  
.....  
</?xml>
```

# Splošna sintaktična pravila dokumentov XML

- ▶ sintaksa v dveh nivojih: splošna in določena z DTD ali shemami XML
- ▶ vsi dokumenti XML se začnejo z deklaracijo
- ▶ sintaksa (splošna):
  - en korenski element,
  - občutljivost na velike/male črke,
  - imena XML: začeti se morajo s črko ali podčrtajem, vsebujejo lahko še številke, pomišljaje in pike,
  - zahtevano pravilno gnezdenje (`< a> < b> < /b> < /a>` in ne `< a> < b> < /a> < /b>` )
  - vsak element mora imeti začetno in končno značko
- ▶ primer dobro tvorjenega (**well-formed**) dokumenta XML:

```
<?xml version = "1.0" encoding = "utf-8" ?>
<student>
  <ime> Marko Hribar </ime>          <letnik> 2 </letnik>
  <vpisna> 63960001 </vpisna>        <naslov> Prekmurska 3, Ljubljana </naslov>
  <ocene>
    <predmet> <naziv>Programiranje</naziv> <ocena>9</ocena> </predmet>
  </ocene>
</student>
```

# Elementi in atributi

- ▶ element: `<student> <ime>"Danko Bananko"</ime></student>`
- ▶ atribut: `<student ime="Danko Bananko"> </student>`
- ▶ kdaj uporabiti atribut in kdaj vgnezden element?
  - je potrebna hierarhija?
  - je potrebna struktura?
  - ali želimo opisati lastnost ali komponento?
  - ali predstavlja vrednost iz dane množice vrednosti?
  - ko zapisujemo identifikator ali enolično ime → običajno atribut

```
<patient name = "Jane Jo Brown">  
...  
</patient>
```



```
<patient>  
  <name>  
    <first> Jane</first>  
    <middle> Jo </middle>  
    <last> Brown </last>  
  </name>  
  ...  
</patient>
```

# Ustreznost dokumenta XML

## ► ločimo dve kategoriji

- **dobra tvorjenost (well-formedness):** dokument je sintaktično pravilno zapisan (gnezdenja, značke, glava)
- **veljavnost (validness):** način podatkov v dokumentu XML je zapisan na pravilen način, kot to določajo pravila za zapis podatkov te vrste (DTD ali shema XML)

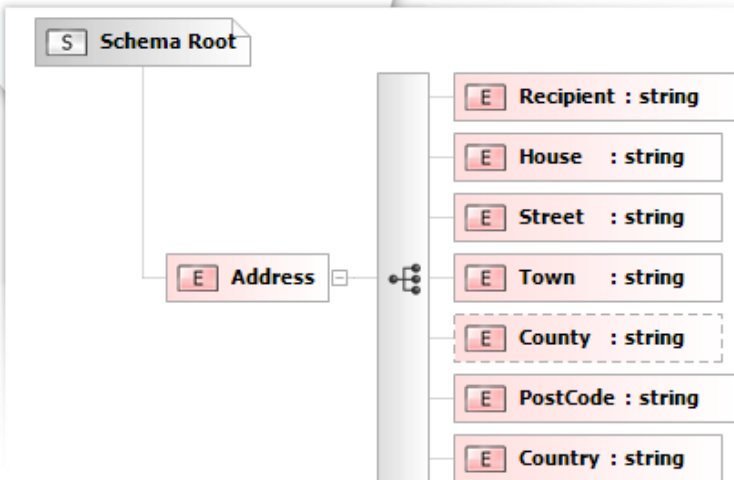
*Vsak veljaven dokument je tudi dobro tvorjen, ni pa vsak dobro tvorjen tudi veljaven.*

# XML

- ▶ XML: uvod in osnovna sintaksa
- ▶ standardizacija oblike dokumenta XML
  - definicija tipa dokumenta (DTD)
  - sheme XML
- ▶ prikaz dokumentov XML
  - prikaz s slogovnimi predlogami CSS
  - prikaz s transformacijami XSLT
  - prikaz z branjem XML DOM
- ▶ razčlenjevalniki XML







# Standardizacija oblike dokumenta XML

# Sintaktična pravila - Definicija tipa dokumenta

- ▶ Document Type Definition (DTD) - množica strukturnih pravil, ki določajo množico elementov in atributov ter kje in kako se lahko pojavijo
- ▶ pravila so deklaracije, ki določajo pravilno strukturo dokumenta XML
- ▶ imamo **notranje** in **zunanje** DTD

- notranje:

```
<!DOCTYPE ime_sheme [  
    ...  
>
```

- zunanje:

```
<!DOCTYPE ime_sheme SYSTEM "datoteka">
```

```
<?xml version="1.0"?><!DOCTYPE books [  
  <!ELEMENT title (#PCDATA)>  
  <!ELEMENT author (#PCDATA)>  
  <!ELEMENT authors (author)+>  
  <!ELEMENT subject (#PCDATA)>  
  <!ATTLIST subject class CDATA "">  
  <!ELEMENT book (title,authors,subject)>  
  <!ATTLIST book  
    bookid CDATA #REQUIRED  
    pubdate CDATA #REQUIRED  
>  
<books name="My books">  
  <book bookid="1" pubdate="03/01/2002">  
    <title>Java Web Services</title>  
    <authors>
```

## Pomembne deklaracije v DTD - element

- ▶ element je podatkovna struktura za shranjevanje informacije
- ▶ deklaracija elementa specificira strukturo ene kategorije elementov
- ▶ deklaracija vsebuje ime elementa, katerega struktura se definira, in specifikacijo strukture
- ▶ **ELEMENT:** definira možne naslednike v drevesni strukturi

```
<!ELEMENT ime_elementa (otrok1[+|*|?], otrok2[+|*|?], ...)>
```

primeri (vozlišča in list):

```
<!ELEMENT memo (from, to, date, re, body)>
```

```
<!ELEMENT person (parent+, age, spouse?, sibling*)>
```

```
<!ELEMENT leaf (#PCDATA)>
```

# Pomembne deklaracije v DTD - atributi

- ▶ atributi zagotavljajo informacijo o elementu
- ▶ atributi so deklarirani ločeno od elementov
- ▶ deklaracija mora vsebovati ime elementa, kateremu pripada, ime atributa, tip atributa in privzeto vrednost
- ▶ **ATTLIST**: definira možne attribute značke

`<!ATTLIST element atribut tip_atr privzeta_vrednost>`

- 10 tipov, privzete vrednosti so lahko podane, #FIXED, #REQUIRED, #IMPLIED

`<!ATTLIST car doors CDATA "4">`

`<!ATTLIST car engine_type CDATA #REQUIRED>`

`<!ATTLIST car price CDATA #IMPLIED>`

`<!ATTLIST car make CDATA #FIXED "Ford">`

če je več deklaracij atributov jih lahko tudi združimo

- uporaba:

`<car doors = "2" engine_type = "V8"> ... </car>`

# Pomembne deklaracije v DTD - entitete

























- ▶ entiteta je simbolična predstavitev informacije
- ▶ entitete so lahko generalne (referenca kjerkoli v dokumentu XML) ali parametrične (referenca samo v DTD)
- ▶ **ENTITY: definira entitete (posebna zaporedja znakov)**
  - `<!ENTITY [%] ime_entitete "vrednost">` (% parametrična entiteta)
  - definicija: `<!ENTITY dld "Danes je lep dan">`
  - uporaba: `&dld;` (primer entitete v HTML: `&nbsp;`;) )
- ▶ zunanja tekstovna entiteta (poglavje, članek, ipd):
  - `<!ENTITY ime_entitete SYSTEM "nahajalisce_datoteke">`

# Pomembne deklaracije v DTD - entitete

- ▶ entitete, ki vsebujejo binarne podatke
  - `<!ENTITY ime SYSTEM "nahajalisce/slika.jpg" NDATA JPEG>`
  - `<!ENTITY slika SYSTEM "disk/slika.jpg" NDATA JPEG>`  
`<!ELEMENT foto EMPTY>`  
`<!NOTATION JPEG SYSTEM "image/jpeg">`  
`<!ATTLIST foto ent ENTITY #REQUIRED>`  
Tole je ena slika: `<foto ent = "slika" />`

# Pomembne deklaracije v DTD – listi drevesa

- ▶ element je lahko vozlišče znotraj drevesa ali pa list drevesa
- ▶ **listi drevesa (poljubno zaporedje znakov)**
- ▶ zaporedja znakov: CDATA in PCDATA - (Parsed) Character Data
- ▶ PCDATA procesira razčlenjevalnik XML, taka zaporedja ne morejo vsebovati znakov, uporabljenih za označevanje
- ▶ uporabiti je potrebno ustrezne reference na znakovne entitete

 &quot;	 &amp;	 &lt;	 &gt;	 &sim;	 &cong;	 &asymp;	 &neq;
 &hArr;	 &loz;	 &spades;	 &clubs;	 &perp;	 &sdot;	 &nbsp;	 &excl;
 &prod;	 &sum;	 &minus;	 &lowast;	 &not;	 &shy;	 &reg;	 &macr;



# Primer DTD

```
<!ELEMENT planes_for_sale (ad+)>
<!ELEMENT ad (year, make, model, color, description, price?, seller, location)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT make (#PCDATA)>
<!ELEMENT model (#PCDATA)>
<!ELEMENT color (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT seller (#PCDATA)>
<!ELEMENT location (city, state)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ATTLIST seller phone CDATA #REQUIRED>
<!ATTLIST seller email CDATA #IMPLIED>
<!ENTITY c "Cessna">
<!ENTITY p "Piper">
<!ENTITY b "Beechcraft">
```

- ▶ preverjanje veljavnosti oblike dokumenta
  - <http://www.xmlvalidation.com/>
  - <http://www.validome.org/grammar/>
  - [http://www.w3schools.com/xml/xml\\_validator.asp](http://www.w3schools.com/xml/xml_validator.asp)

```
<planes_for_sale>
  <ad>
    <year> 1977 </year>
    <make> Cessna </year>
    <model> Skyhawk </model>
    <description>New paint, nearly new interior, ...</description>
    <seller phone="555-222-3333">Skyway Aircraft</seller>
    <location>
      <city> Rapid City </city>
      <state> South Dakota </state>
    </location>
  </ad>

  <ad>
    .....
  </ad>
</planes_for_sale>
```





# XML

- ▶ XML: uvod in osnovna sintaksa
- ▶ standardizacija oblike dokumenta XML
  - definicija tipa dokumenta (DTD)
  - sheme XML
- ▶ prikaz dokumentov XML
  - prikaz s slogovnimi predlogami CSS
  - prikaz s transformacijami XSLT
  - prikaz z branjem XML DOM
- ▶ razčlenjevalniki XML



# Sintaktična pravila - Sheme XML

- ▶ DTD – imajo nekaj pomanjkljivosti:
  - ima drugačno sintakso kot XML – se ne da analizirati z razčlenjevalniki XML, nejasno zaradi dveh sintaks
  - malo podatkovnih tipov (10 za attribute), ne omogoča omejitev glede podatkov (npr. numerični podatki in datum specificirani kot tekst)
- ▶ sheme XML so ena izmed alternativ, podpora s strani W3C
  - v obliki XML, lahko analiziramo z razčlenjevalniki XML
  - veliko tipov (44), močnejša sintaksa za definiranje strukture dokumenta in podatkov
- ▶ sheme so definirane s korenskim elementom

```
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema" ... >
```

  - xmlns atribut določa imenski prostor
  - niz za dvopičjem določa predpono vsakega elementa iz tega imenskega prostora
  - niz za enačajem določa imenski prostor oziroma jezik značk shema

# Imenski prostori

- ▶ XML omogoča kreiranje lastnih elementov – kolizija elementov z istim imenom
- ▶ sistem, ki omogoča razlikovanje pomenov značk z istim imenom in z različnimi pomeni (npr. <table> kot tabela ali kos pohištva)
- ▶ imenski prostor naslovimo in ga označimo z nizom, ki ga uporabimo kot predpono značkam → <element\_name xmlns[:prefix] = URI>
- ▶ en imenski prostor je lahko brez predpone, to predstavlja privzet imenski prostor
- ▶ primer definiranja in uporabe imenskih prostorov za element file

```
<text:directory
  xmlns:text = "urn:deitel:textInfo"
  xmlns:image = "urn:deitel:imageInfo">
  <text:file filename = "book.xml">
    <text:description>A book list</text:description>
  </text:file>
  <image:file filename = "funny.jpg">
    <image:description>A funny picture</image:description>
    <image:size width = "200" height = "100" />
  </image:file>
</text:directory>
```



# Uporaba sheme v dokumentu XML

- ▶ primer definicije:

```
<xsd:schema
```

```
  xmlns:xsd = "http://www.w3.org/2001/XMLSchema"
```

```
  targetNamespace = "http://cs.uccs.edu/planeSchema"
```

```
  xmlns = "http://cs.uccs.edu/planeSchema"
```

```
  elementFormDefault = "qualified">
```

imenski prostor sheme

imenski prostor ciljnega dokumenta

privzeti imenski prostor za ta dokument

uporabljamo gnezdenje značk (elementi so gnezdeni)

- ▶ primer definiranja dokumenta XML z uporabo sheme:

```
<planes
```

```
  xmlns = "http://cs.uccs.edu/planeSchema"
```

```
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
```

```
  xsi:schemaLocation = "http://cs.uccs.edu/planeSchema planes.xsd" >
```

korenski element sheme

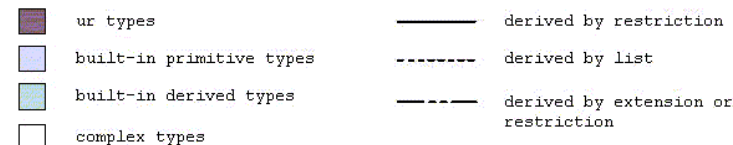
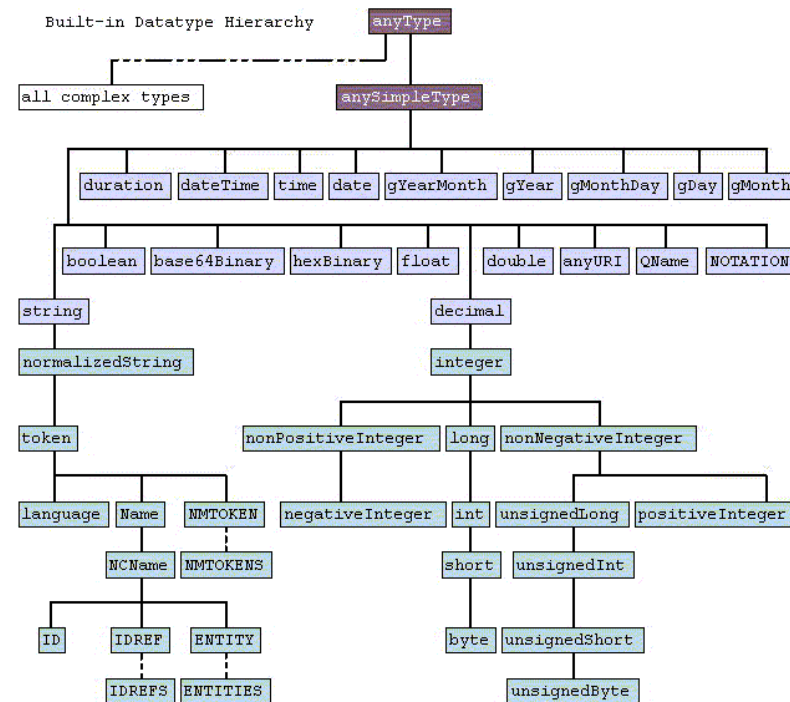
imenski prostor tega dokumenta

imenski prostor primerka sheme

imenski prostor in lokacija sheme!!!

# Podatkovni tipi v shemah XML

- ▶ **44 osnovnih podatkovnih tipov** (string, Boolean, float, byte, decimal, ...):
  - 19 primitivnih,
  - 25 izpeljanih
- ▶ uporabniški tipi:
  - **preprosti** (dodajanje omejitev, ...),
  - **kompleksni** (razširjanje z elementi, ...)



# Podatkovni tipi v shemah XML

## ▸ osnovni tipi:

definicija elementa:

```
<xsd:element name = "engine" type = "xsd:string" />
```

instanca sheme, v kateri je definiran element:

```
<engine> inline six cilinder fuel injected </engine>
```

definiramo lahko tudi privzete vrednosti:

```
<xsd:element name = "engine" type = "xsd:string"  
  default = "fuel injected V6" />
```

definiramo lahko tudi fiksne vrednosti:

```
<xsd:element name = "engine" type = "xsd:string"  
  fixed = "petrol" />
```

- preprosti tip **simpleType** ima lahko omejitve (maxLength, minInclusive, ...)

```
<xsd:simpleType name = "middleName" >  
  <xsd:restriction base = "xsd:string" ><xsd:maxLength value = "20" />  
  </xsd:restriction>  
</xsd:simpleType>
```

# Podatkovni tipi v shemah XML

- ▶ definiranje **sestavljenih (kompleksnih)** tipov:
  - element **complexType**, sledi ime tipa
  - urejen vrstni red elementov zahtevamo z elementom `sequence`, neurejen vrstni red pa z `all`

```
<xsd:complexType name = "sports_car" >  
  <xsd:sequence>  
    <xsd:element name = "make" type = "xsd:string" />  
    <xsd:element name = "model " type = "xsd:string" />  
    <xsd:element name = "engine" type = "xsd:string" />  
    <xsd:element name = "year" type = "xsd:string" />  
  </xsd:sequence>  
</xsd:complexType>
```

- določimo lahko tudi število pojavitev posameznega elementa z `minOccurs` in `maxOccurs`

# Podatkovni tipi v shemah XML

## ► definiranje atributov elementa:

- element, ki vsebuje atribut, mora biti tipa **complexType**, ima lahko preprosto vsebino (list drevesa, extension) ali pa urejeno/neurejeno zaporedje (notranje vozlišče, complexType)

```
<xsd:element name = "avto" >
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base = "xsd:string">
        <xsd:attribute name = "barva" type = "xsd:string" />
        <xsd:attribute name = "motor" type = "xsd:string" use = "required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType> </xsd:element>
```

- `<avto barva = "srebrna" motor="V2">Spaček</avto>`
- atributi so privzeto opcijski, posebej je potrebno določiti, da so zahtevani
- atributi morajo biti definirani na koncu tipa





# Primer DTD in XML sheme

<!ELEMENT planes\_for\_sale (ad+)>

<!ELEMENT ad (year, make, model, color, description, price?, seller, location)>

<!ELEMENT year (#PCDATA)>

<!ELEMENT make (#PCDATA)>

<!ELEMENT model (#PCDATA)>

<!ELEMENT color (#PCDATA)>

<!ELEMENT description (#PCDATA)>

<!ELEMENT price (#PCDATA)>

<!ELEMENT seller (#PCDATA)>

<!ELEMENT location (city, state)>

<!ELEMENT city (#PCDATA)>

<!ELEMENT state (#PCDATA)>

<!ATTLIST seller phone CDATA #REQUIRED>

<!ATTLIST seller email CDATA #IMPLIED>

<!ENTITY c "Cessna">

<!ENTITY p "Piper">

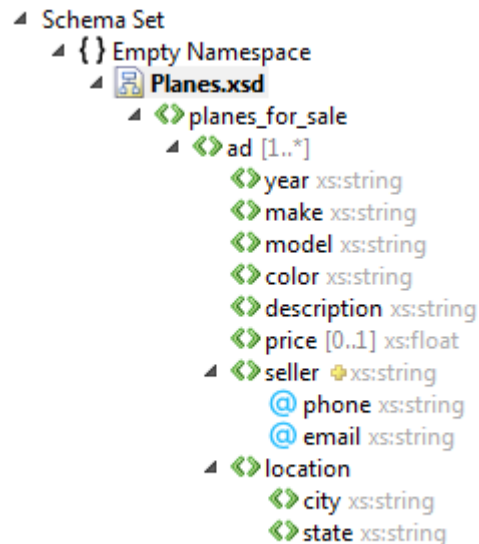
<!ENTITY b "Beechcraft">

```
<planes_for_sale>
  <ad>
    <year> 1977 </year>
    <make> Cessna </year>
    <model> Skyhawk </model>
    <description>New paint, nearly new interior, ...</description>
    <seller phone="555-222-3333">Skyway Aircraft</seller>
    <location>
      <city> Rapid City </city>
      <state> South Dakota </state>
    </location>
  </ad>

  <ad>
    .....
  </ad>
</planes_for_sale>
```



```
<?xml version="1.0"?>
<xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema'
  elementFormDefault='qualified'>
  <xs:element name='planes_for_sale'>
    <xs:complexType>
      <xs:sequence>
        <xs:element name='ad' maxOccurs='unbounded'>
          <xs:complexType>
            <xs:sequence>
              <xs:element name='year' type='xs:string'/?>
              <xs:element name='make' type='xs:string'/?>
              <xs:element name='model' type='xs:string'/?>
              <xs:element name='color' type='xs:string'/?>
              <xs:element name='description' type='xs:string'/?>
              <xs:element name='price' type='xs:float' minOccurs='0' maxOccurs='1'/?>
              <xs:element name='seller'>
                <xs:complexType>
                  <xs:simpleContent>
                    <xs:extension base='xs:string'>
                      <xs:attribute name='phone' type='xs:string' use='required'/?>
                      <xs:attribute name='email' type='xs:string'/?>
                    </xs:extension>
                  </xs:simpleContent>
                </xs:complexType>
              </xs:element>
              <xs:element name='location'>
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name='city' type='xs:string'/?>
                    <xs:element name='state' type='xs:string'/?>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```





```
<?xml version="1.0"?>
<xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema'
  elementFormDefault='qualified'>
  <xs:element name='year' type='xs:string'/>
  <xs:element name='make' type='xs:string'/>
  <xs:element name='model' type='xs:string'/>
  <xs:element name='color' type='xs:string'/>
  <xs:element name='description' type='xs:string'/>
  <xs:element name='price' type='xs:float'/>
  <xs:element name='city' type='xs:string'/>
  <xs:element name='state' type='xs:string'/>

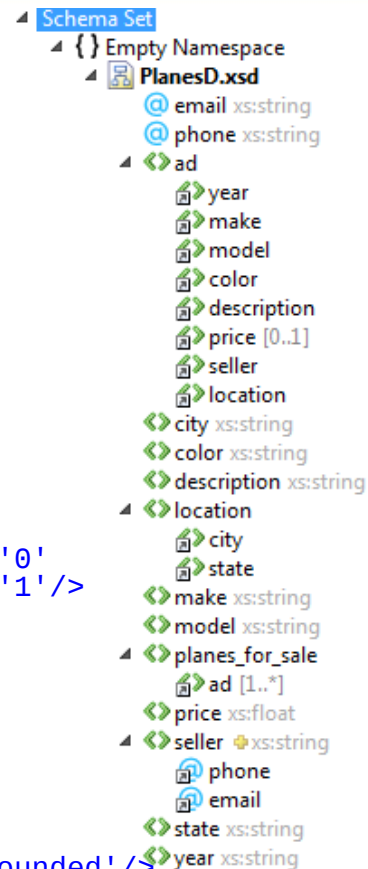
  <xs:attribute name='phone' type='xs:string'/>
  <xs:attribute name='email' type='xs:string'/>

  <xs:element name='seller'>
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base='xs:string'>
          <xs:attribute ref='phone' use='required'/>
          <xs:attribute ref='email'/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>

  <xs:element name='location'>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref='city'/>
        <xs:element ref='state'/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

```
<xs:element name='ad'>
  <xs:complexType>
    <xs:sequence>
      <xs:element ref='year'/>
      <xs:element ref='make'/>
      <xs:element ref='model'/>
      <xs:element ref='color'/>
      <xs:element ref='description'/>
      <xs:element ref='price' minOccurs='0'
        maxOccurs='1'/>
      <xs:element ref='seller'/>
      <xs:element ref='location'/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name='planes_for_sale'>
  <xs:complexType>
    <xs:sequence>
      <xs:element ref='ad' maxOccurs='unbounded'/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```



# XML

- ▶ XML: uvod in osnovna sintaksa
- ▶ standardizacija oblike dokumenta XML
  - definicija tipa dokumenta (DTD)
  - sheme XML
- ▶ prikaz dokumentov XML
  - prikaz s slogovnimi predlogami CSS
  - prikaz s transformacijami XSLT
  - prikaz z branjem XML DOM
- ▶ razčlenjevalniki XML



# Prikaz dokumentov XML s CSS

- ▶ brskalnik ne ve kako formatirati poljuben dokument XML, vsebuje lahko poljubne značke, zato prikaže drevesno strukturo dokumenta
- ▶ zagotoviti moramo slogovne pole
- ▶ oblikovanje vključimo z direktivo

```
<?xml-stylesheet type = "text/css" href = "mydoc.css"?>
```

- ▶ primer CSS

```
ad { display: block; margin-top: 15px; color: blue;}  
year, make, model { color: red; font-size: 16pt;}  
color {display: block; margin-left: 20px; font-size: 12pt;}  
description {display: block; margin-left: 20px; font-size: 12pt;}  
seller { display: block; margin-left: 15px; font-size: 14pt;}  
location {display: block; margin-left: 40px; }  
city {font-size: 12pt;}  
state {font-size: 12pt;}
```

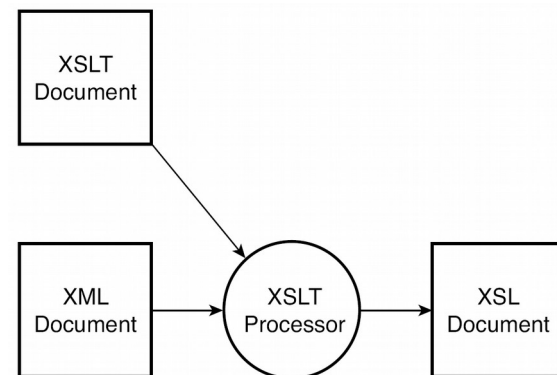
# XML

- ▶ XML: uvod in osnovna sintaksa
- ▶ standardizacija oblike dokumenta XML
  - definicija tipa dokumenta (DTD)
  - sheme XML
- ▶ prikaz dokumentov XML
  - prikaz s slogovnimi predlogami CSS
  - prikaz s transformacijami XSL
  - prikaz z branjem XML DOM
- ▶ razčlenjevalniki XML



# Slogovne predloge XSLT

- ▶ XSL = XSLT + XPath
- ▶ Lastnosti eXtensible Stylesheet Language Transformations (XSLT):
  - definirane s standardnim XML
  - možen direkten dostop do elementa z uporabo XPath in obdelava elementov v poljubnem vrstnem redu
- ▶ XSLT - transformacija XML dokumenta v poljubno drugačno obliko (HTML, tekstovni, XML dokument).
- ▶ Tehnično gledano:
  - dokument XSLT - igra vlogo programa
  - dokument XML - igra vlogo vhoda v program
  - dokument XSL - igra vlogo izhoda programa
- ▶ dokument XML oblikujemo z uporabo XSLT z direktivo:



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

# Slogovne predloge XSLT

- ▶ slogovna predloga XSLT je dokument XML, ki vsebuje element stylesheet
- ▶ predloga vsebuje kombinacijo "posebnih" značk za predloge in navadnih izpisov (HTML), privzeti imenski prostor je prostor XHTML

```
<xsl:stylesheet xmlns:xsl = http://www.w3.org/1999/XSL/Format  
                xmlns = "http://www.w3.org/1999/xhtml">
```

- ▶ pomembne značke:

```
<xsl:template match = "XPath">
```

pozicionira program na pot v dokumentu,  
predloga mora vsebovati vsaj en element template  
za celoten dokument

npr. 

```
<xsl:template match = "/">
```

```
<xsl:value-of select = "XPath" />
```

izpiše vrednost elementa na poti

```
<xsl:for-each select = "XPath">
```

iterira preko elementov na poti





# Slogovne predloge XSLT – Primer

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="/pozdrav-svetu"> <!-- ujemanje na korenski element -->
    <html>
      <head>
        <title>Primer XSL/Transformacij</title>
        <style>
          h1 { color: #ff0055;}
          div {
            font-size: 2em;
            color: #abcdef;
          }
        </style>
      </head>
      <body>
        <h1>
          <xsl:value-of select="pozdrav"/>
        </h1>
        <div>
          posilja
          <i><xsl:value-of select="pozdravljaec"/></i>
        </div>
      </body>    </html>
    </xsl:template>
  </xsl:stylesheet>
```

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="Pozdrav.xsl"?>
<pozdrav-svetu>
  <pozdravljaec>XSLT programer</pozdravljaec>
  <pozdrav>Pozdravljen, svet!!!</pozdrav>
</pozdrav-svetu>
```



# Slogovne predloge XSLT – Primer (XML)

```
<?xml version = "1.0"?>
<?xml-stylesheet type = "text/xsl" href = "Studenti-table.xsl"?>
<studenti fakulteta="FRI">
  <student vpisna="66550022">
    <ime>Danko</ime> <priimek>Bananko</priimek>
    <naslov>Ulica banan 5, Ljubljana</naslov>
  </student>
  <student vpisna="66550017">
    <ime>Danko</ime> <priimek>Bananko</priimek>
    <naslov>Ulica banan 6, Ljubljana</naslov>
  </student>
  <student vpisna="66440003">
    <ime>Maruška</ime> <priimek>Hruška</priimek>
    <naslov>Ulica hrušk 20, Hruševo</naslov>
  </student>
  <student vpisna="66440001">
    <ime>Anka</ime> <priimek>Zaspanka</priimek>
    <naslov>Zaspana ulica 1, Spalci</naslov>
  </student>
  <student vpisna="66440054">
    <ime>Vinko</ime> <priimek>Potepinko</priimek>
    <naslov>Ulica potepov 20, Potepi</naslov>
  </student>
</studenti>
```



# Slogovne predloge XSLT – Primer (XSL)

```
<?xml version = "1.0"?>
<xsl:stylesheet version = "1.0" xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
  <xsl:output method = "html" doctype-system = "about:legacy-compat" />
  <xsl:template match = "/"> <!-- ujemanje na korenski element -->
    <html>
      <head> <title>Studenti</title> <meta charset = "utf-8"/>
        <link rel = "stylesheet" type = "text/css" href = "style.css"/>
      </head>
      <body>
        <p> <xsl:value-of select = "/studenti/@fakulteta"/> </p>
        <table> <thead>
          <td> Vpisna številka</td> <td> Ime</td> <td> Priimek</td> <td> Naslov</td>
        </thead>
        <xsl:for-each select = "/studenti/student">
          <xsl:sort select="priimek" /> <xsl:sort select="@vpisna" />
          <tr> <td> <xsl:value-of select = "@vpisna"/></td>
            <td><xsl:value-of select = "ime"/></td>
            <td><xsl:value-of select = "priimek"/></td>
            <td><xsl:value-of select = "naslov"/></td> </tr>
        </xsl:for-each>
      </table> </body> </html>
    </xsl:template>
  </xsl:stylesheet>
```



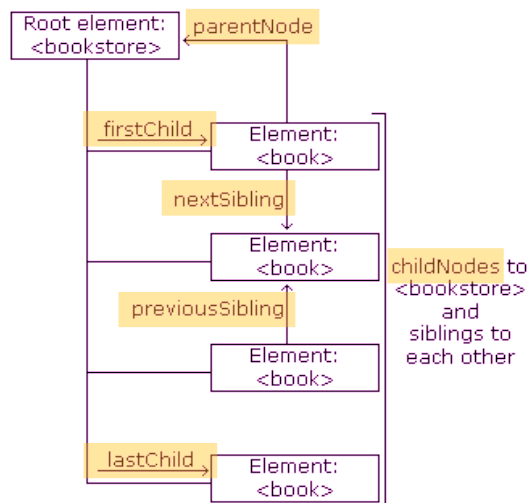
# XML

- ▶ XML: uvod in osnovna sintaksa
- ▶ standardizacija oblike dokumenta XML
  - definicija tipa dokumenta (DTD)
  - sheme XML
- ▶ prikaz dokumentov XML
  - prikaz s slogovnimi predlogami CSS
  - prikaz s transformacijami XSLT
  - prikaz z branjem XML DOM
- ▶ razčlenjevalniki XML



# XML DOM

- ▶ brskalnik prebrano datoteko XML predstavi z drevesno predstavitevjo, imenovano XML DOM
- ▶ elementi XML so vozlišča, med njimi so podobne relacije kot v HTML DOM
- ▶ do vozlišč lahko dostopamo in jih spreminjamo
- ▶ dokument viden pred procesiranjem – hitro ugotovimo (ne)pravilno tvorjenost dokumenta (well-formedness)



## Uporabne metode

- ▶ `x.nodeName`
- ▶ `x.nodeValue`
- ▶ `x.parentNode`
- ▶ `x.childNodes`
- ▶ `x.attributes`
- ▶ `x.getAttribute`
- ▶ `x.getElementsByTagName(name)`
- ▶ `x.appendChild(node)`
- ▶ `x.removeChild(node)`



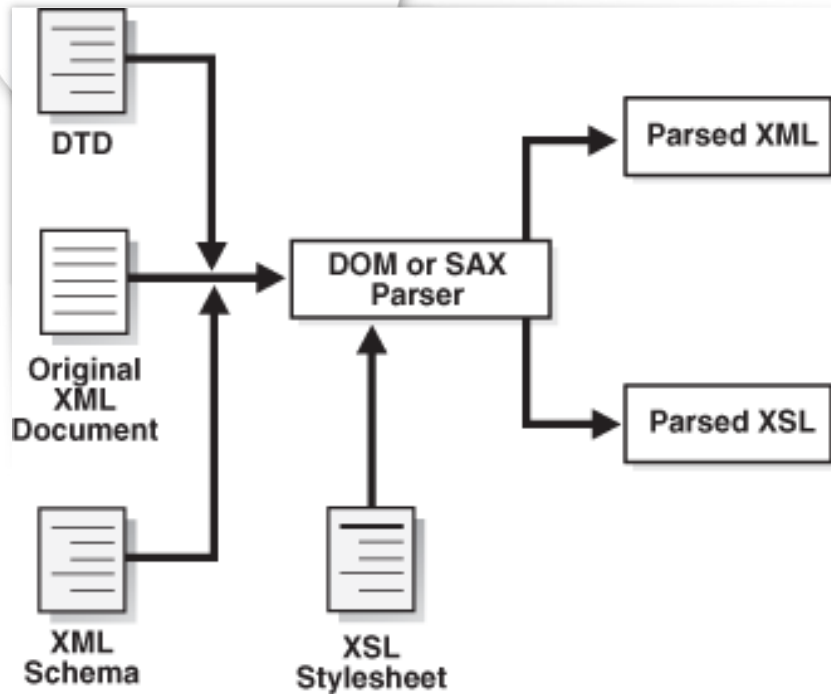
# XML DOM-Primer (HTML+JS prikaz XML vsebine)

```
...<script type = "text/javascript" src = "readxml.js" > </script>
...
<body>
  <div id="besedilo"></div>
  <input type="button" onclick="prejsnji()" value="<<" />
  <input type="button" onclick="naslednji()" value=">>" />
  <script type = "text/javascript" >
    xmlDoc =loadXMLDoc("StudentiP.xml")
    x = xmlDoc.getElementsByTagName("student");
    var i = 0;
    function prikaziStudenta() {
      vprisna = (x[i].getAttribute("vpisna"));
      ime = (x[i].getElementsByTagName("ime")[0].childNodes[0].nodeValue);
      priimek = (x[i].getElementsByTagName("priimek")[0].childNodes[0].nodeValue);
      naslov = (x[i].getElementsByTagName("naslov")[0].childNodes[0].nodeValue);
      txt= "Vpisna st.: " + vprisna + "<br> Ime: " + ime + "<br> Priimek: " +
          priimek + "<br> Naslov: " + naslov;
      document.getElementById("besedilo").innerHTML= txt;
    }
    prikaziStudenta();
    function naslednji() { if (i < x.length-1) {i++; prikaziStudenta();}}
    function prejsnji() { if (i>0) {i--; prikaziStudenta();}}
  </script>
</body>
</html>
```

# XML

- ▶ XML: uvod in osnovna sintaksa
- ▶ standardizacija oblike dokumenta XML
  - definicija tipa dokumenta (DTD)
  - sheme XML
- ▶ prikaz dokumentov XML
  - prikaz s slogovnimi predlogami CSS
  - prikaz s transformacijami XSLT
  - prikaz z branjem XML DOM
- ▶ razčlenjevalniki XML





# Razčlenjevalniki XML, uporaba XML



# Razčlenjevalniki XML

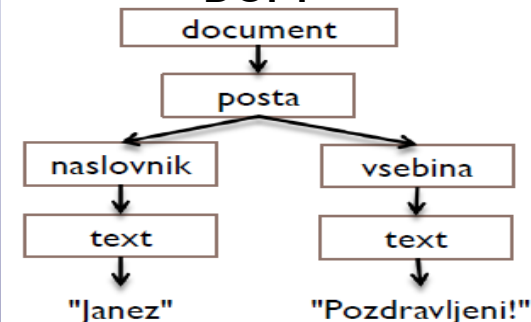
## ► Procesiranje dokumentov XML:

- preverjanje pravilne tvorjenosti dokumentov (well-formedness)
- zamenjava entitet z definicijami,
- vstavljanje privzetih vrednosti iz DTD in shem XML,
- dva načina procesiranja: pristop z DOM in pristop SAX (Simple API for XML)
  - DOM: dokument se razčleni v drevesno strukturo, po tem lahko delamo obhode po elementih,
  - SAX: sprotno branje dokumenta, ob branju značk, atributov itd. se prožijo dogodki

### DOKUMENT

```
<posta>  
  <naslovnik> Janez </naslovnik>  
  <vsebina> Pozdravljeni! </vsebina>  
</posta>
```

### DOM



# Razčlenjevanje s pristopom SAX

- ▶ Simple API for XML (SAX)
- ▶ razčlenjevanje s pristopom SAX – dogodkovno procesiranje
- ▶ razčlenjevalnik pregleduje dokument XML od začetka do konca
- ▶ ko prepozna sintaktično strukturo signalizira dogodek s klicanjem rokovalnika dogodkov za dano strukturo
- ▶ sintaktična struktura vsebuje značko za odprtje, attribute, tekst in značko za zaprtje

## DOKUMENT

```
<posta>
  <naslovnik> Janez </naslovnik>
  <vsebina> Pozdravljeni! </vsebina>
</posta>
```

## SAX

- dogodki pri branju
  - start document
  - start element: posta
  - start element: naslovnik
  - characters: Janez
  - ...



# Razčlenjevanje s pristopom DOM ali SAX ?

- ▶ Prednosti pristopa z DOM
  - omogoča večkratni in naključen dostop do istih elementov
  - lažja prerazporeditev elementov, če je dostopen cel dokument
  - preveri pravilno tvorjenost dokumenta
- ▶ Slabosti pristopa z DOM
  - potrebuje več spomina
  - je bolj počasen