XML:n syntaksi

HOHDE

- Peruskäsitteitä
- Rakennepalaset
  - elementit
  - leipäteksti
  - attribuutit
- Nimiavaruudet
- Dokumentin rakenteen määrittely
  - Document Type Definition
  - XML Schema

Kirjainherkkyys

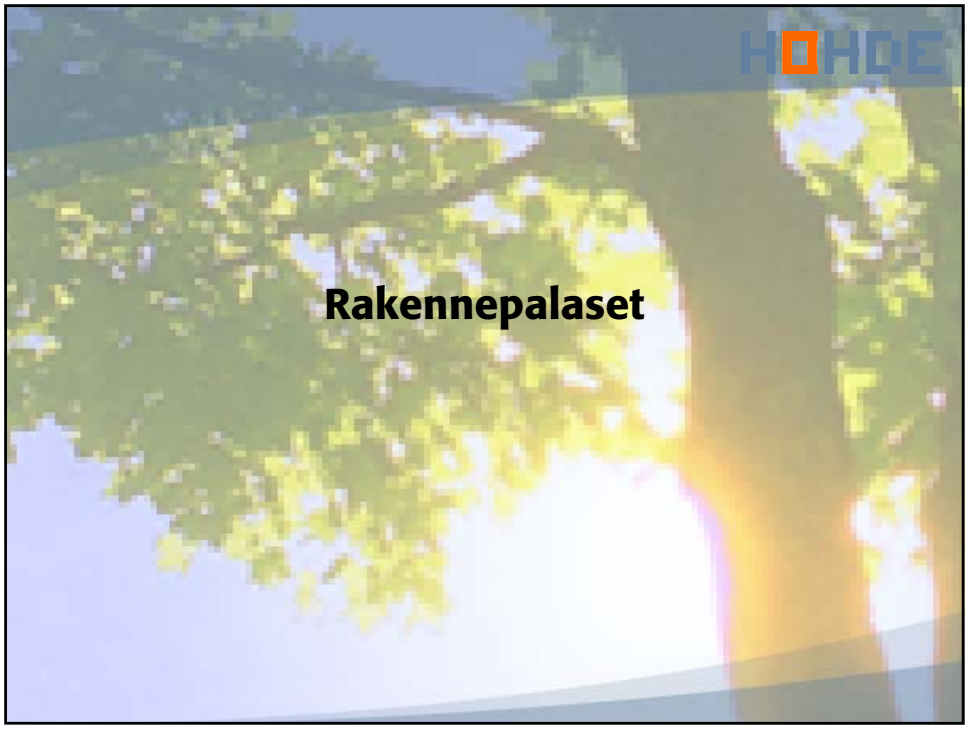
- XML on kirjainherkkä kieli (case sensitive)
  - isot ja pienet kirjaimet ovat eri merkkejä

    `<element/>`, `<Element/>` ja `<ELEMENT/>`

    ovat eri elementtejä
- HTML ei ole kirjainherkkää
- Suositeltavaa on valita yksi tyyli ja käyttää sitä koko dokumentissa
  - yleisimpiä ovat <elementname/> ja <ElementName/>
  - XML-pohjainen standardi tai suositus saattaa määrittää käytetyn tavan
    - noudata sitä

Unicode

- Unicode on merkistöstandardi
  - käytössä XML:ssa
  - tarkoituksena tukea kaikkia maapallon kieliä ja muita notaatioita (matemaattiset kaavat yms.)
  - useita eri määrittelyjä
    - osa merkkialueesta vielä käyttämättä
  - sisältää tuen kaikille tärkeimmille aakkostoille
  - kaikki modernin käyttöjärjestelmät tukevat Unicodea
- XML käyttää oletusarvoisesti merkkien koodaamiseen UTF-8-esitystapaa
  - latinalaiset aakkoset esitetään kahdeksalla bitillä, erikoisemmat merkit pitempiä
  - XML-dokumentissa voidaan kuitenkin valita mikä tahansa esitystapa
- Käytännössä nykyiset tekstieditorit ja muut ohjelmat piilottavat merkistön käyttäjältä



XML:n rakennepalaset

- XML-dokumentti voi sisältää seuraavia rakenteita:
  - elementit – muodostavat dokumentin rungon ja rakenteen
  - attribuutit – antavat lisämääreitä elementeille
  - leipäteksti – elementtien sisällä olevaa tekstiä
  - kommentit – kommentteja ihmisille tai koneilta väliaikaisesti piilotettuja osioita
  - käsittelyohjeet – ohjeita dokumentin koneelliseen käsittelyyn
  - entiteetit – erikoismerkkien esittämiseen ja dokumenttien liittämiseen toisiinsa

Elementit

- Dokumentti koostuu elementeistä (element)
  - jokaisella elementillä on nimi
  - elementti kirjoitetaan kulmasulkujen (< ja >) sisään
  - elementillä on yleensä alku- ja loppuosa
    - loppuosa merkitään </
    - alku- ja loppuosien tulee täsmätä

```
<element>...</element>
```

- Puhutaan myös tageista (tag)
  - tarkoittaa välillä elementtiä, välillä elementin alku- tai loppuosaa (start/end tag)
  - tällä kurssilla puhutaan elementeistä

Elementtien nimet

- Elementtien nimille on tiettyjä sääntöjä
  - nimi voi alkaa ainoastaan kirjaimella tai alaviivalla
  - nimessä on sallittu käyttää kirjaimia, numeroita, ala- ja tavuviivoja ja pisteitä
- Vaikka XML on unicodea, niin kannattaa välttää skandinaavisia merkkejä ja muita erikoiskirjaimia
  - ohjelmat eivät välttämättä osaa tulkita oikein
  - kaikki ohjelmat eivät myöskään hallitse kaikkia koodaustapoja ja erikoismerkit voivat tällöin aiheuttaa ongelmia

Elementtien sisäkkäisyys

- Elementit voivat sisältää toisia elementtejä tai leipätekstiä
- Sisempiä elementtejä kutsutaan lapsielementeiksi (child element) ja ulompaa isäelementiksi (parent element)
  - dokumentin ulointa elementtiä kutsutaan juurielementiksi (root element)
- Sisäkkäisyydestä muodostuu dokumentin rakenne
  - sisäkkäisyys ei voi mennä ristiin

```
<element>...<ali>...</ali>...</element>
```

Tyhjät elementit

- Kaikkien elementtien ei tarvitse olla kaksiosaisia
- Yksiosaisia elementtejä kutsutaan tyhjiksi elementeiksi (empty element)
  - merkitään lisäämällä kauttaviiva elementin nimen perään

```
<empty-element/>
```

  - yllä esitetty elementti voidaan kirjoittaa myös

```
<empty-element></empty-element>
```

- Huom! HTML:ssä ei tehdä eroa tyhjien ja normaalien elementtien välillä
  - HTML-taustaisten ihmisten yksi yleisimmistä perusvirheistä on unohtaa tyhjien elementtien merkintä

Leipäteksti

- Elementit voivat sisältää leipätekstiä sisällään
  - lapsielementit ja leipäteksti voivat vaihdella vapaasti

```
<element>Hello<World/>!</element>
```

- Teknisesti, jokainen erillinen leipätekstisirpale sijoitetaan omaan leipätekstisolmuun XML-puussa
  - yllä olevassa esimerkissä on kaikkiaan neljä solmua:
    - elementti "element"
    - teksti "Hello"
    - elementti "World"
    - teksti "!"

CDATA-alue

- Leipätekstissä ei voida käyttää XML:n syntaksissa varattuja merkkejä <, > ja &
  - näitä varten on olemassa korvausmerkinnät (<, > ja &), mutta aina niidenkään käyttö ei ole mahdollista
- Leipätekstin sijaan voidaan käyttää CDATA-aluetta (CDATA section), joka voi sisältää mitä tahansa merkkejä
  - alue aloitetaan merkeillä <![CDATA[ja päätetään merkkeihin]]>
  - Mikäli alueessa on merkit]]>, se täytyy jakaa kahteen osaan: <![CDATA[]]><![CDATA[]]>
- Dokumentin käsittelyn kannalta CDATA-alue rinnastetaan täysin leipätekstiin

Tyhjät merkit

- Tyhjiksi merkeiksi (white spaces) luetaan välilyönnit, tabulaattorit, rivinvaihdot yms.
- XML-dokumentti yleensä tulostetaan siistiin muotoon
  - jokainen elementti omalla rivillään
  - tällöin elementtien väliin voi syntyä tyhjiä merkkejä sisältäviä leipätekstisirpaleita
    - XML-käsittelijän voi ohjata hävittämään pelkästään tyhjistä merkeistä koostuvat leipätekstisirpaleet
- Tyhjät merkit, erityisesti niiden ilmestyminen ja katoaminen, aiheuttavat useasti harmaita hiuksia dokumenttien käsittelyssä
  - törmännette tähän ongelmaan XSLT:n parissa

Attribuutit

- Elementteihin voidaan lisätä tietoa attribuuteilla
- Attribuutti on avain-arvopari
  - attribuutin nimeä vastaa sille annettu arvo
  - yhdessä elementissä voi olla useita attribuutteja
    - vain yksi samanniminen sallitaan, muuten määrittely ei ole yksikäsitteinen

```
<empty-element attr="value" attr2="value2"/>
```


Attribuuttien syntaksi

- Attribuuttien nimiä koskevat samat säännöt kuin elementtienkin nimiä
  - Attribuutit erotetaan elementin nimestä ja toisistaan välilyönneillä
- Attribuutilla pitää aina olla arvo
  - tyhjä arvo merkitään pelkillä lainausmerkeillä
- Attribuutin arvon täytyy olla aina lainausmerkeissä
  - lainausmerkin ilmaisemisessa käytetään korvausmerkintää
  - arvon sisältö voi olla mitä tahansa tekstiä ja entiteettejä

Attribuutti vs. lapsielementti

- Attribuutti voidaan aina korvata lapsielementillä
  - attribuutin nimestä saadaan elementin nimi
  - attribuutin arvosta saadaan elementin sisältö
    - esitetään leipätekstinä

```
<element attr="value"/>

<element>
  <attr>value</attr>
</element>
```

- Valinta attribuutin ja lapsielementin välillä on makukysymys
  - lapsielementin käyttö mahdollistaa myöhemmän laajentamisen ja sallii useita arvoja
  - attribuutti on yleensä helpompi ohjelmoijille ja vähentää ongelmia tyhjien merkkien kanssa
    - näitä syntyy automaattisesti elementtien ympärille, kun dokumentti tulostetaan sisennettynä

Kommentit

- Aina dokumentista ei selviä kaikki oleellinen ihmislukijalle
- Dokumenttia voidaan selventää kommenteilla
  - konekäsittelijä jättää kommentit huomiotta
- Kommentti alkaa merkeillä `<!--` ja päättyy merkkeihin `-->`
  - kommentit eivät voi olla sisäkkäisiä
    - kommentti päättyy ensimmäiseen -->-merkintään

```
<!-- This is a comment -->
```

- Kommentti on näppärä tapa sulkea osa dokumentista konekäsittelyn ulkopuolelle
  - tällöin on huomattava, että ensimmäinen kommentin sulkumerkki päättää kommentin (vaikka kommentti olisi aloitettu useamman kerran)

Entiteetit

- Dokumentissa voidaan käyttää tiettyä joukkoa entiteettejä (entity)
  - merkitään &entity;
  - lyhennys- tai korvausmerkintä tietylle merkille tai merkkijonolle
  - myös mikä tahansa merkki voidaan ilmoittaa entiteettinä
    - entiteetti alkaa risuaidalla (#) ja tämän jälkeen merkin Unicode-koodi ilmoitetaan heksadesimaalisena
    - esimerkiksi — on listapallukka
  - <, >, &, " ja ' ovat korvattavissa entiteeteillä (<, >, &, " ja ')
    - muut entiteetit täytyy itse määritellä (DTD:ssa)

Käsittelyohjeet

- Käsittelyohjeet vastaavat kommentteja koneelliselle käsittelijälle
  - niillä voidaan antaa ohjeita tietylle koneelliselle käsittelijälle
  - käsittelijä voi jättää ohjeet huomiottakin
  - dokumentin käsittely ei saisi riippua ohjeiden täsmällisestä tulkinnasta

```
<?program attr="value"?>
```

- Käsittelyohje on kuten tyhjä elementti, mutta se alkaa merkeillä <? ja loppuu merkkeihin ?>
- Käsittelyohjeen nimi viittaa sitä tulkitsevaan käsittelijään
  - nimi `xml` on varattu standardin käyttöön
- Ohjeet välitetään normaaleina attribuutteina

XML-määrittely

- XML-dokumentin alussa voi olla erityinen XML-määrittely
  - sisältää ohjeita XML-jäsentimelle
  - ei näy suoranaisesti itse XML-dokumentti käsittelevälle sovellukselle
- XML-määrittelyssä voidaan asettaa muun muassa käytetty merkistökoodaus ja dokumentin kieli

```
<?xml encoding="iso-8859-1"?>
```

- XML-määrittely ei ole käsittelyohje, vaikka se näyttääkin siltä
  - se voi sijaita vain tiedoston alussa ja sen sisältämää tietoa ei voi lukea XML-jäsentimeltä

XML:n versiointi

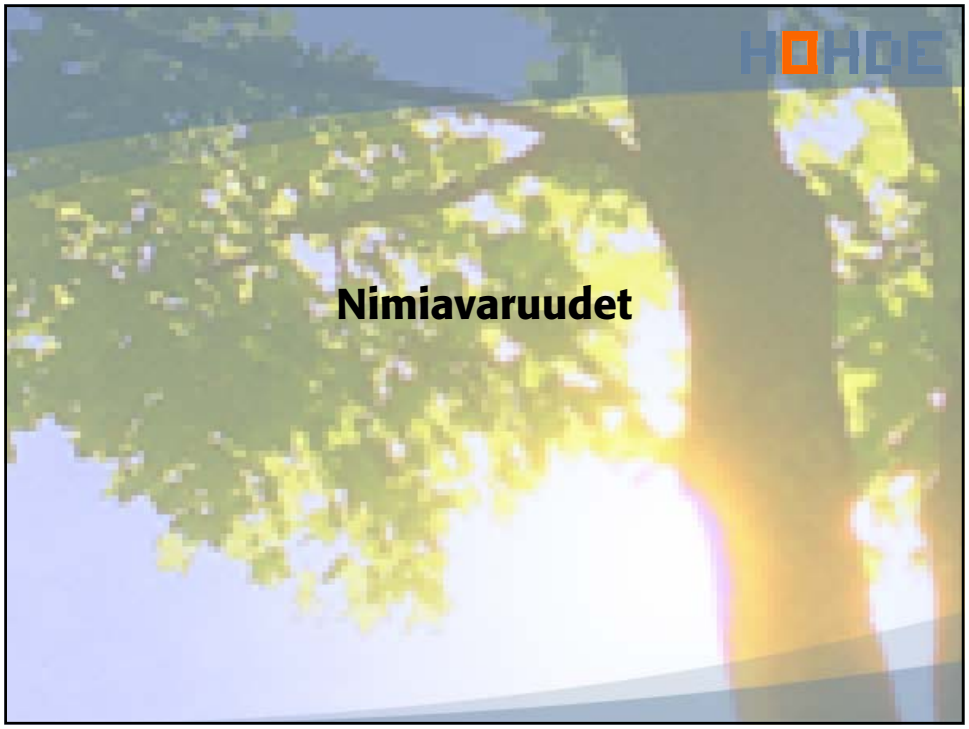
- XML-dokumentin alussa pitää kertoa käytetyn standardin versionumero
  - tämäkin kerrotaan XML-määrittelyssä
  - nykyinen versio on 1.0

```
<?xml version="1.0" standalone="yes" encoding="UTF-8"?>
```

  - standalone määrittää, että dokumentti ei riipu toisista tiedostoista (DTD)
    - vaihtoehdot joko "yes" tai "no"
  - World Wide Web Consortium (W3C) on määrittelemässä XML 1.1 -suositusta
    - syntaksi ei ole muuttumassa
    - hienosäätöä merkistöjen yms. suhteen

Hyvämuotoinen XML

- XML-dokumentti on hyvämuotoista, kun
  - dokumentti alkaa XML-määrittelyllä
  - isäelementeillä on alku- ja loppuosa
  - tyhjät elementit on merkitty loppukauttaviivalla
  - elementit ovat sisäkkäin, eivät ristikkäin
  - dokumentissa on yksi elementti, jonka lapsia kaikki muut elementit ovat
    - tätä elementtiä kutsutaan dokumentin juurielementiksi tai juureksi
  - dokumentissa ei käytetä merkkejä < ja & kuin elementtien ja entiteettien alussa
- Jos dokumentti ei täyty em. vaatimuksia, se ei ole virallisesti XML:a



Nimiavaruudet

- On mahdollista kirjoittaa dokumentteja, jotka käyttävät elementtejä useista XML-kieliopeista
  - esimerkiksi XHTML-dokumenttiin otetaan mukaan matemaattiset kaavat MathML-kielestä
- Tällöin voi tulla tilanne, että eri kieliopit määrittävät samat elementit
  - sovellus ei voi olla varma, minkä kieliopin mukaan elementtejä pitäisi tulkita
- Ongelma on ratkaistu nimiavaruuksilla
  - jokainen kielioppi kuuluu omaan nimiavaruuteen
  - dokumentin elementit kytketään nimiavaruuksiin, jolloin elementin käsittely on yksikäsitteistä

Nimiavaruuden käyttö

- Nimiavaruus määritellään aina jollakin URI:lla, ts. WWW-osoitteella
- URI:t ovat hankalia, jos niitä pitäisi toistaa jokaisen elementin kohdalla
  - on kehitetty lyhennysmerkintä
  - dokumentin alussa nimiavaruus-URI:in sidotaan joku lyhenne, jota käytetään elementtien nimen edessä, esimerkiksi: <fo:block/> tai <xsl:apply-templates/>
- Myös attribuuteilla voi olla nimiavaruus
  - oletusarvoisesti attribuutti kuuluu samaan nimiavaruuteen kuin elementti, mutta myös attribuutin nimen eteen voi liittää nimiavaruuden

Nimiavaruuden määrittely

- Käytetään attribuuttia `xmlns`

```
<hohde:element xmlns:hohde="http://www.hohde.com/">
```

- Määritellään nimiavaruus ja käytetään sitä elementissä
  - tässä tapauksessa nimiavaruuden nimi on ”hohde”
  - erotetaan elementistä kaksoispisteellä
- Yleensä nimiavaruudet määritetään dokumentin juurielementissä
  - käytettävissä tämän jälkeen kaikissa elementeissä
- Nimiavaruuden URI:n ei tarvitse viitata mihinkään tiettyyn dokumenttiin
  - se toimii paremminkin nimiavaruuden tunnistimena

Nimiavaruuden perintä

- Kaikki elementit eivät välttämättä määrittele nimiavaruutta
  - dokumentissa voidaan määritellä näille elementeille oma nimiavaruus, perusnimiavaruus
  - käytetään attribuuttia `xmlns` juurielementissä
    - attribuutille ei anneta nimiavaruusmäärettä

```
<element xmlns="http://www.hohde.com/"/>
```

Paikalliset elementit

- Jos perusnimiavaruutta ei määritellä, kaikki nimiavaruudettomat elementit ovat paikallisia
  - niihin ei voida sitoa samanlaista yleistä merkitystä kuin nimiavaruudellisiin elementteihin
  - aina tätä ei edes haluta
- Jos dokumentin elementtiä ei haluta sitoa URI:in, niin tällöin xmlns-attribuutin arvoksi täytyy laittaa tyhjä merkkijono

```
<element xmlns=""/>
```


Dokumentin rakenteen määrittely

Dokumentin rakenteen määrittely

- XML-dokumentin sallittu rakenne voidaan määritellä
  - dokumentti validoidaan määrittelyä vastaan
  - mikäli dokumentti on määritellyissä rajoissa, se hyväksytään
- Rakenteen formaali määrittely helpottaa dokumenttia tulkitsevien sovellusten laadintaa
- Määrittely muodostaa rungon dokumentin semantiikalle
  - dokumentin kielioppi syntyy määrittelyssä
  - dokumentin sisältö määrittää kuitenkin dokumentin tarkoituksen

Document Type Definition

- Document Type Definition (DTD) on määrittely, jolla luodaan XML-dokumentin rakenne
  - DTD määrittää käytettävissä olevat elementit, niiden attribuutit ja leipätekstin mahdolliset paikat
  - elementtien järjestystä ja sisäkkäisyyttä voidaan säädellä
  - attribuutit voidaan määrittää pakollisiksi tai vapaaehtoisiksi, ja niille voidaan antaa oletusarvot
- DTD:n syntaksi periytyy SGML:sta ja se ei ole XML-merkintöjen mukaista
  - osittain tästä syystä DTD on korvautumassa XML Schemalla
    - suurempi syy on DTD:n ilmaisuvoiman puutteet

Elementtien määrittely

- DTD:ssa määritetään jokainen dokumentissa esiintyvä elementti
  - elementtiin sidotaan tietty määrä attribuutteja
    - osa voi olla pakollisia
    - attribuuttien arvoille voi olla rajoituksia
- Jokaiselle elementille luetellaan mahdolliset lapsielementit
  - lapsielementtien järjestys, lukumäärä ja toistettavuus määritellään samalla
- Jokainen elementti määritellään vain kerran
  - toisin sanoen elementin rakenne ei voi riippua sen sijainnista dokumentissa

Lapsielementtien määrittely

- Lapsielementtien nimet luetellaan suluissa elementin nimen jälkeen
  - nimet erotetaan toisistaan pilkuilla
  - lapsien järjestys sitoo dokumenttia
  - jos lapsielementti on vapaaehtoinen, sen perään merkitään ?
  - jos lapsielementti toistuu, sen perään merkitään +
  - jos lapsielementti toistuu ja se on vapaaehtoinen, sen perään merkitään *
- Lapsielementit voivat olla vaihtoehtoisia
  - erotetaan pystyviivalla | pilkun sijasta
- Lapsielementtejä voidaan ryhmitellä suluilla
  - lapsielementtiryhmät voivat olla vapaa- ja vaihtoehtoisia ja toistua kuten normaalitkin lapsielementit

Muut lapsimäärittelyt

- Lapseksi voidaan merkitä #PCDATA
  - parsed character data
  - tekstiä, jossa ei ole elementtimäärittelyjä (=leipätekstiä)
- Tyhjien elementtien lapsien tilalle merkitään EMPTY

Attribuuttien määrittely

- Jokainen elementin attribuutti määritellään erikseen
  - sama attribuuttimäärittely kelpaa vain yhteen elementtiin
- XML tukee useita erilaisia attribuuttityyppejä, alla yleisimmät
  - CDATA – merkkipohjainen
    - käytetyin attribuuttityyppi
  - Enumeroitu – yksi arvo muutamasta vaihtoehdosta
    - esimerkiksi pata, hertta, ruutu ja risti
  - ID – ainutkertainen nimi dokumentissa
    - arvo saa esiintyä ainoastaan kerran dokumentissa kaikkien dokumentin ID-attribuuttien arvoissa
  - IDREF, IDREFS – viittauksia ID-attribuuttiin
  - ENTITY, ENTITIES – DTD:ssä määritelty entiteetti

Attribuuttien arvojen määrittely

- XML:lle ei voi kertoa attribuutin arvon tarkkaa tyyppiä kuten ohjelmointikielissä
  - tyyppi on aina tekstiä
  - tulkitsevan ohjelman tulee tehdä tarkistukset
- Attribuutin arvolle voidaan määrätä rajoitteita
  - #REQUIRED – arvo tulee olla aina dokumentissa
  - #IMPLIED – jos arvoa ei ole, käsittelijä päättää sen itse
  - #FIXED – arvo määrätään DTD:ssä, dokumentti ei voi määrittää elementtiä
  - oletusarvo

DTD:n syntaksi

- Jokainen elementti määritellään seuraavasti:

```
<!ELEMENT element-name (child-elements)>
```

  - esimerkiksi

```
<!ELEMENT document (head, body)>
```

- Tyhjä elementti vastaavasti

```
<!ELEMENT element-name EMPTY>
```

- Attribuutti sidotaan elementtiin

```
<!ATTLIST element-name attribute type value>
```

- Useita samaan elementtiin liittyviä attribuutteja voidaan niputtaa samaan määrittelyyn

```
<!ATTLIST element-name attr1 type value
                       attr2 type value
                       attr3 type value>
```

DTD-esimerkki

```
<!DOCTYPE document [
  <!ELEMENT document (head, ingress, body)>
  <!ELEMENT head (#PCDATA)>
  <!ELEMENT ingress (#PCDATA)>
  <!ELEMENT body ((paragraph | picture)*)>
  <!ELEMENT paragraph (#PCDATA)>
  <!ELEMENT picture EMPTY>
     <!ATTLIST picture url CDATA #REQUIRED>
     <!ATTLIST picture align
       (left | right | middle) "left">
     <!ATTLIST picture text CDATA #IMPLIED>
]>
```


DTD-esimerkkejä (1/2)

- Lapsielementtien määrittelyä

```
(sub) - yksi lapsi
(sub, bus) - kaksi lasta
(sub, sub) - kaksi samaa lasta
(sub, bus, sub, bus) - vuorotellen
(sub?, bus) - vapaaehtoinen lapsi
(sub+, bus*) - toistuvia lapsia
(sub | bus) - joko sub tai bus
(sub+ | bus) - toisto ja vapaaehtoisuus
(sub | bus?) - joko sub, bus tai ei mitään
((sub, bus) | usb) – sub ja bus tai vain usb
((sub+ | bus), usb) – useita subeja tai yksi
bus, lopuksi usb
(#PCDATA) – vain leipätekstiä
(#PCDATA | p | i)* – elementtejä tai tekstiä
```

DTD-esimerkkejä (2/2)

- Attribuuttien määrittelyä

```
<!ATTLIST e attr CDATA #IMPLIED>
elementin <e> attribuutti attr, joka sisältää
merkkipohjaista dataa ja ei ole pakollinen

<!ATTLIST e attr ID #REQUIRED>
pakollinen attribuutti, jonka arvon tulee olla
uniikki

<!ATTLIST e attr CDATA #FIXED ”value”>
attribuutti, jonka arvoa ei voi muuttaa

<!ATTLIST e attr (a | b | c) ”c”)
attribuutti, jonka arvo voi olla a, b tai c ja se
on oletusarvoisesti c
```


DTD:n liittäminen dokumenttiin

- DTD voi olla joko dokumentin sisällä tai siihen viitataan dokumentista

```
<!DOCTYPE root-element [
        DTD-määritykset
]>
```

- Ulkoiseen DTD:in viitataan seuraavalla syntaksilla

```
<!DOCTYPE root-element SYSTEM "url">
```

  - URL määrää, mistä DTD haetaan
- Osa DTD:sta on ns. julkisia, jolloin niille on annettu erityinen nimi

```
<!DOCTYPE root-element PUBLIC "name" "url">
```

  - nimen rakentamisessa on erilliset käytännöt, joihin ei nyt syvennytä
  - standardit ja suositukset käyttävät yleensä julkisia DTD-määrityksiä

DTD:n ongelmat

- DTD periytyy SGML:sta ja on auttamattoman vanhentunut XML:n tarpeisiin
  - syntaksi poikkeaa XML:sta
  - ei tukea nimiavaruuksille
  - ei kunnollista tukea tietotyypeille
  - kielen rakenteen määrittely osittain rajoittunutta

XML Schema

- XML Schema on kieli, jolla määritellään muita XML-pohjaisia kieliä
  - XML Schema -määrittelyillä luodaan kielen rakenteet
    - määritellään käytettävissä olevat elementit ja attribuutit
    - asetetaan rajoituksia elementtien sisäkkäisyydelle ja peräkkäisyydelle
    - määritetään attribuuttien arvoilla tyypit ja mahdolliset raja-arvot
    - lisätään attribuuteille oletusarvot tai todetaan tietyt attribuutit pakollisiksi
  - rakenteen pohjalta syntyy dokumentin sisältö ja semantiikka
    - määritelty rakenne mahdollistaa dokumenttien tarkistuksen ja helpottaa dokumenttien koneellista käsittelyä
- XML Scheman tarkoitus on korvata DTD

Elementtien määrittely

- Elementit määritellään <element>-elementillä:

```
<element name="document"/>
```

  - määriteltiin elementti <document>
  - attribuuteilla voidaan lisätä määreitä elementtiin, esimerkiksi säätää elementin sisältö pelkäksi tekstiksi

```
<element name="document" type="string"/>
```


Rakenteen määrittely

- Elementit voidaan määritellä sisäkkäin, jolloin samalla syntyy rakenne:

```
<element name="document">
     <complexType>
            <sequence>
                   <element name="title" type="string"/>
            </sequence>
     </complexType>
</element>
```

- Monimutkaisempia rakenteita (toistot, ehdollisuudet) varten on erillisiä elementtejä:

```
<element name="chapter">
     <complexType>
            <choice minOccurs="0" maxOccurs="1">
                   <element .../>
                   <element .../>
            </choice>
     </complexType>
</element>
```

Elementtien uusiokäyttö

- Edellä määritellyt elementit ovat paikallisia, eli käytettävissä vain isä-elementin sisällä
- Elementtijoukko voidaan määritellä myös erikseen (global) ja tähän elementtijoukkoon voidaan viitata toisen elementin sisältä
  - vähennetään elementtimäärittelyjen lukumäärää
  - määrittelyssä käytetään <complexType>-elementtiä ja attribuuttia name:

```
<complexType name="titleType">
  <element name="title" type="string"/>
  ...
</complexType>
```

- Tähän määrittelyyn voidaan viitata käyttämällä type- tai ref-attribuuttia:

```
<element name="chapter" type="target:titleType"/>
<element ref="target:titleType/>
```


Attribuutit

- Attribuutit määritellään elementillä <attribute>:

```
<element name="document">
    <complexType>
          <attribute name="title"
           type="string"/>
          ...
    </complextType>
</element>
```

Muut ominaisuudet

- XML Schema on hyvin laaja määrittely ja siihen kannattaa tutustua tarkemmin lukemalla oppikirjaa:
  - elementtien määrän säätäminen
  - ehdollisuudet
  - attribuuttien tietotyypit ja arvoalueen asettaminen
  - attribuuttien globaali määrittely
  - oletusarvot
  - ryhmittelyt
  - yms.

Validointi

- Validoinnissa verrataan XML-dokumenttia sen DTD-määrittelyyn tai Schemaan
  - dokumentin jokainen elementti, attribuutti ja leipätekstisirpale tarkastetaan
  - mikäli DTD/Schema ei salli em. osasta tai joku vaadittu osanen uupuu, dokumentti ei ole DTD:n/Scheman mukainen ja validointi epäonnistuu
- Validointi on vapaaehtoista
  - XML-dokumentin ei ole edes pakko viitata DTD:iin/Schemaan
  - XML-prosessori suorittaa validoinnin
    - prosessorille voidaan ilmoittaa, että validointia ei saa suorittaa tai se on pakko suorittaa
- Validoinnilla sovellus voi varmistaa, että sisään luettu dokumentti on ainakin syntaktisesti oikein
  - validointi ei löydä semanttisia tai loogisia virheitä
- Validointi voi tapahtua myös XML Schemaa vasten


Kysymyksiä? Kommentteja?