Introduction to SGML: **Standard Generalized Markup Language**

Tomaž Erjavec
Jožef Stefan Institute
Ljubljana
Slovenia

EUROLAN'97
Summer School in Corpus Linguistics
Tusnad - Romania
13-26 July 1997

Overview of the lecture

1. Preliminaries: what, why, when, how, who…
2. The standard
3. Software, concluding remarks

What you can expect from the course:

1. an understanding of SGML
2. good basics for understanding TEI, CES

Sources used for the slides:

- C.M. Sperberg-McQueen and Lou Burnard:
  Gentle Introduction to SGML.
  In *Guidelines for Electronic Text Encoding and Interchange* (C.M. Sperberg-McQueen, Lou Burnard eds.), Chicago and Oxford, 1994.

  URL: *http://etext.virginia.edu/TEI.html*
- Eve Maler and Jeanne el Andaloussi:
  Developing SGML DTDs: From Text to Model to Markup.
  Prentice Hall, New Jersey, 1996.

What is SGML?

From *The Gentle Introduction*:

“SGML is an international standard for the definition of device-independent, system-independent methods of representing texts in electronic form”

“SGML is a metalanguage, that is, a means of formally describing a language, in this case, a markup language”

- **markup** (encoding): making explicit an interpretation of text
- **markup language**: a set of markup conventions used together for encoding texts; it must specify what markup is allowed, what markup is required, how markup is to be distinguished from text, and what the markup means.

The ISO standard & its History

International Organization for Standardization,
`ISO 8879: Information processing---Text and office systems---Standard Generalized Markup Language (SGML)`, ([Geneva]: ISO, 1986).

As such, it is *not proprietary*.

History:

In 1969, Charles Goldfarb leads an **IBM** research project on integrated law office information systems. With E. Mosher and R. Lorie he invented the Generalized Markup Language (GML) as a means of allowing text editing, formatting, and information retrieval subsystems to share documents.

The first working draft of the SGML standard was published in 1980 by **ANSI**. By 1983, the sixth working draft is recommended as an industry standard (GCA 101-1983). Major adopters included US IRS and DoD.

A draft **ISO** standard was published in October 1985, and was adopted by the Office of Official Publications of the EU. Another year of review and comment resulted in the final text, which was published in record time after approval (**ISO 8879:1986**).

Characteristics of SGML

Descriptive Markup

Markup codes **categorize** parts of a document; they do not tell what processing is to be carried out at particular points in a document (procedural markup).

E.g.:

- "the following item is a paragraph"
- "skip down one line, move 5 quads right"

Document Types

Documents are regarded as having types, and these are expressed by document type definitions (**DTD**), which enforce markup for that document type.

Data Independence

SGML encoded documents should be transportable from one hardware and software environment to another without loss of information: platforms differ in character sets, file-naming conventions, interpretation of bytes...

SGML provides a general purpose mechanism for **string substitution**, that is, a simple machine-independent way of stating that a particular string of characters in the document should be replaced by some other string when the document is processed.

Why is SGML used?

- managing large amounts of valuable, (predominantly) textual data
- interchange: people, platforms, time
- multiple exploitation of texts

For example:

- industry: technical manuals
- publishers
- law firms
- libraries
- WWW
- language data for "language technologies":
  - **corpora**
  - dictionaries, lexicons
  - ...

Important initiatives:

- Text Encoding Initiative (**TEI**)
- EAGLES and other EU projects

An example of SGML markup

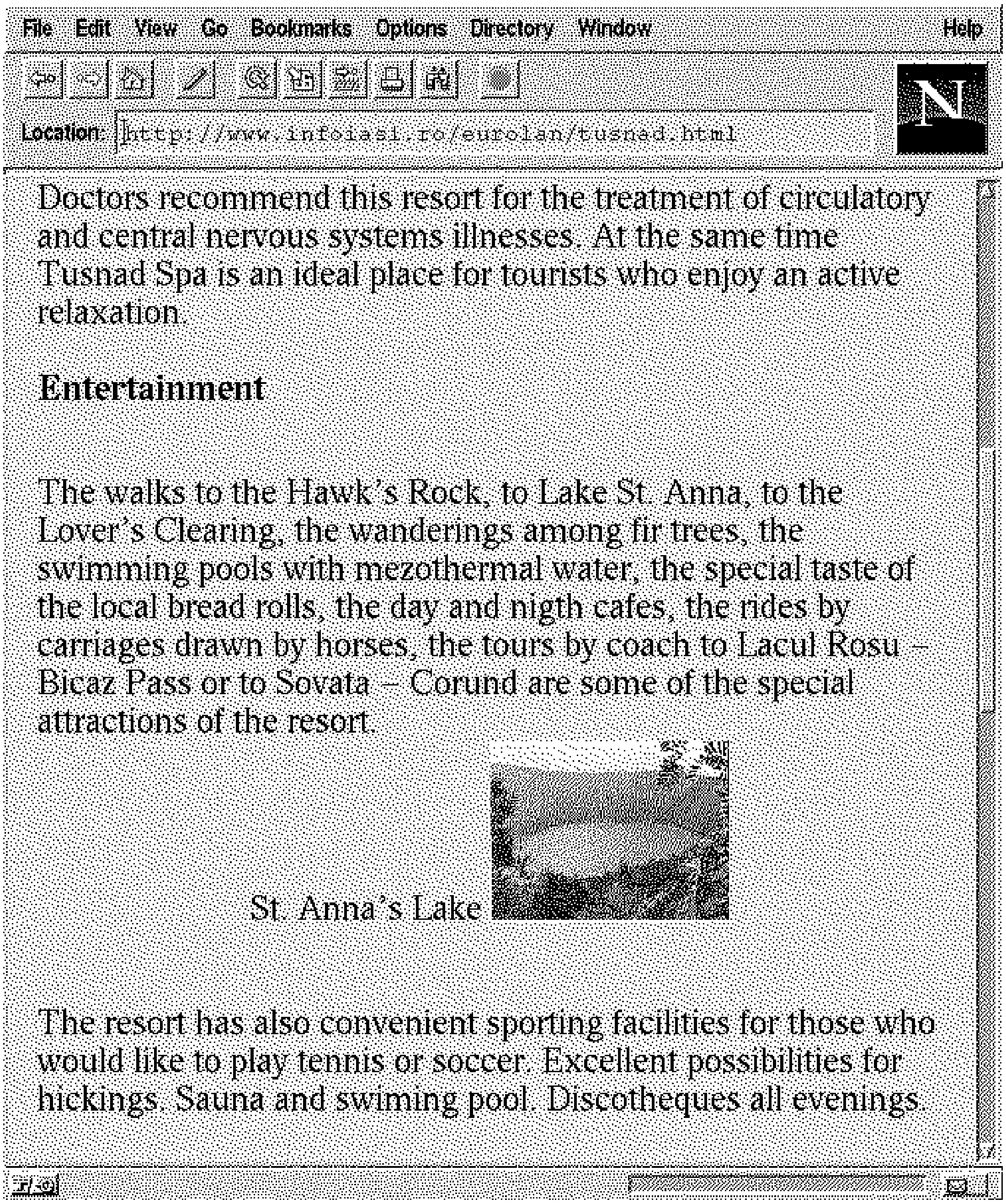
- simple ASCII
- markup distinguished from text
- markup encloses text
- markup contains other markup

```
<anthology>
  <poem><title>The SICK ROSE</title>
     <stanza>
          <line>O Rose thou art sick.</line>
          <line>The invisible worm,</line>
          <line>That flies in the night</line>
          <line>In the howling storm:</line>
     </stanza>
     <stanza>
          <line>Has found out thy bed</line>
          <line>Of crimson joy:</line>
          <line>And his dark secret love</line>
          <line>Does thy life destroy.</line>
     </stanza>
  </poem>
          <!-- more poems go here    -->
</anthology>
```

Another example: HTML

```
<P><BR>
Doctors recommend this resort for the treatment of
circulatory and central nervous systems illnesses. At
the same time Tusnad Spa is an ideal place for
tourists who enjoy an active relaxation.
<P>
<H2>Entertainment</H2>
The walks to the Hawk’s Rock, to Lake St. Anna, to
the Lover’s Clearing, the wanderings among fir trees,
the swimming pools with mezothermal water, the
special taste of the local bread rolls, the day and
night cafes, the rides by carriages drawn by horses,
the tours by coach to Lacul Rosu - Bicaz Pass or to
Sovata - Corund are some of the special attractions
of the resort.
<BR>
<CENTER>St. Anna’s Lake <IMG SRC="t51.gif"></CENTER>
<BR>
<BR>
The resort has also convenient sporting facilities
for those who would like to play tennis or soccer.
Excellent possibilities for hickings. Sauna and
swimming pool.  Discotheques all evenings.
```

HTML as seen on a browser



And Yet Another: CES

```
 <div id="Oro.1.2" type=chapter n=1>
   <head>1</head>
   <p id="Oro.1.2.2">
     <s id="Oro.1.2.2.1">&Icirc;ntr-o zi
     senin&abreve; &scedil;i friguroas&abreve; de
     aprilie, pe c&acirc;nd ceasurile b&abreve;teau
     ora treisprezece, <name type="person">Winston
     Smith</name>, cu b&abreve;rbia
     &icirc;nfundat&abreve; &icirc;n piept pentru a
     sc&abreve;pa de v&acirc;ntul care-l lua pe sus,
     se strecur&abreve; iute prin u&scedil;ile de
     sticl&abreve; ale <name type="place">Blocului
     Victoria</name>, de&scedil;i nu destul de
     repede pentru a &icirc;mpiedica un v&acirc;rtej
     de praf &scedil;i nisip s&abreve;
     p&abreve;trund&abreve; o dat&abreve; cu el.
     </s>
   </p>
...
 </div>
```

What constitutes a SGML document?

1. The **document instance**:
   - contains data (text)
   - contains markup
   - must contain/refer to a DTD
2. The **document type definition**:
   - defines the structure of a document
   - may be stored externally to a document
3. The **SGML declaration**:
   - usually common to all documents in an SGML installation
   - if it isn't a part of the document, a default is used

The SGML declaration

Defines the concrete SGML syntax used on a system:

- **delimiter roles**
  what is markup, what is text? e.g. `STAGO` = `<`
- **reserved names**
  what are the strings that mean something to SGML?
  e.g. `ELEMENT` used in element definitions
- **disallowed characters**
  which characters are forbidden to appear?
  e.g. control characters (`^L, ^G`)
- **naming rules**
  what characters can be used in markup names?
  Case-sensitivity of markup. e.g. it defines that `POEM` and `Poem` are the same tag
- **quantities**
  max. length of documents, markup names,...
  e.g. max. length of GI is 32
- **allowed SGML features**
  which features are in force?
  e.g. `DATATAG, OMITTAG, FORMAL`

The SGML declaration usually used is (something close to) the **Reference Concrete Syntax**.

The TEI SGML declaration

```
<!SGML "ISO 8879:1986"
-- TEI SGML declaration for local processing with SGMLS:
         extended capacity points
         full ASCII character set instead of ISO 646 subset
         ^Z (26) defined as end-of-file
         Namelen 32, Litlen 512, otherwise as for TEI interchange
   made 16 Aug 1990, rev. 9 Mar 1991 by CMSMcQ
        increased LITLEN to 2048, GRPCNT to  100 (LB Jul 93) --
CHARSET
BASESET  "ISO 646-1983//CHARSET
          International Reference Version (IRV)//ESC 2/5 4/0"
DESCSET  0       9     UNUSED
         9       1     9
         10      1     10
         11      2     UNUSED
         13      1     13
         14     12     UNUSED
         26      1     UNUSED   -- eof --
         27      5     UNUSED
         32     95     32
        127      1     UNUSED
BASESET  "ISO Registration Number 100//CHARSET
          ECMA-94 Right Part of Latin Alphabet Nr. 1//ESC 2/13 4/1"
DESCSET  128    32     UNUSED
         160    95     32
         255    1      UNUSED
--
CAPACITY PUBLIC "ISO 8879:1986//CAPACITY Reference//EN"
--
CAPACITY  SGMLREF
TOTALCAP    10000000
ENTCAP        500000
ENTCHCAP      500000
ELEMCAP       500000
GRPCAP       1000000
EXGRPCAP      500000
EXNMCAP       500000
ATTCAP        500000
ATTCHCAP      500000
AVGRPCAP      500000
```

```
NOTCAP       500000
NOTCHCAP     500000
IDCAP        500000
IDREFCAP     500000
MAPCAP       500000
LKSETCAP     500000
LKNMCAP      500000

SCOPE    DOCUMENT
SYNTAX
SHUNCHAR CONTROLS 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
                  16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
                  127 255
BASESET  "ISO 646-1983//CHARSET
          International Reference Version (IRV)//ESC 2/5 4/0"
DESCSET   0     128     0
FUNCTION RE     13
         RS     10
         SPACE  32
         TAB    SEPCHAR 9
     --  EOF    SEPCHAR 26 --
NAMING   LCNMSTRT ""
         UCNMSTRT ""
         LCNMCHAR "-."
         UCNMCHAR "-."
         NAMECASE GENERAL YES
                  ENTITY  NO
DELIM    GENERAL SGMLREF
         SHORTREF SGMLREF
NAMES    SGMLREF
QUANTITY SGMLREF
         ENTLVL   32
         NAMELEN  32
         LITLEN  2048
         GRPCNT  100
         GRPGTCNT  180
         TAGLVL   24
FEATURES
MINIMIZE DATATAG NO     OMITTAG  YES    RANK     NO     SHORTTAG YES
LINK     SIMPLE  NO     IMPLICIT NO     EXPLICIT NO
OTHER    CONCUR  NO     SUBDOC   YES 10 FORMAL   NO
APPINFO NONE>
```

Document Type Definition

Each SGML document must refer to a DTD.

The Document Type Definition consists of

- the **formal part**

  formally specifies what markup can appear in the document type and in what relation to other markup and data

- the **documentation**

  specifies what the defined elements **mean**:

  `<date>Joe</date>` can be formally OK, but is nonsense.

The formal DTD consists of:

- **ELEMENT** declarations
- **ATTRIBUTE** declarations
- **ENTITY** specifications

Element declarations

```
<!ELEMENT anthology      - -  (poem+)>
```

Is composed of:

`<!`	Markup declaration open delimiter
`ELEMENT`	declaration type keyword
`anthology`	**generic identifier**
`- -`	omitted tag minimization
`(poem+)`	**content model**
`>`	Markup declaration close delimiter

Some more examples:

```
<!ELEMENT anthology      - -  (poem+)>
<!ELEMENT poem           - -  (title?, stanza+)>
<!ELEMENT title          - O  (#PCDATA) >
<!ELEMENT stanza         - O  (line+)   >
<!ELEMENT line           - -  (#PCDATA) >
```

The document according to the declarations:

```
<anthology>
   <poem>
     <stanza>
          <line>The first stanza.</line>
     <stanza>
          <line>Of a sad song</line>
          <line>this was.</line>
  </poem>
</anthology>
```

Model Group Operators

- `(` open bracket for grouping
- `)` close bracket
- `,` follows
- `|` or
- `?` maybe
- `*` repeated 0 or more times
- `+` repeated once or more times

What to watch out for

- **Content model ambiguity**:

  ```
  <!ELEMENT a   - - ((b, c)|(b, d))>
  ```

- **Mixed content**

  carriage return = record end (**RE**)

  In the DTD:

  ```
  <!ELEMENT listitem  - - (#PCDATA, para+)>
  <!ELEMENT para     - - (#PCDATA)>
  ```

  In the document:

  ```
  <listitem>
  Oranges
  <para>Oranges are a lovely fruit.</para>
  <para>They grow in warm climates.</para>
  </listitem>
  ```

Some further twists

- A more complex poem type:
```
<!ELEMENT poem - O
          (title?,
          ( (line+)
          | (refrain?, (stanza, refrain?)+ ) ))>
```

- **Model Groups**:
```
<!ELEMENT (note | variant) - - (#PCDATA)>
```

- **EMPTY content**:
```
<!ELEMENT poemref - O EMPTY >
```

- **ANY content**:
```
<!ELEMENT memo - - ANY>
```

- **Inclusion / Exclusion Exceptions**:
```
<!ELEMENT poem - O
          (title?, (stanza+ | couplet+ | line+))
          +(note | variant) >

<!ELEMENT title - O
          (#PCDATA)
          -(note | variant) >

<!ELEMENT (note | variant) - -
          (#PCDATA)
          -(note | variant) >
```

ATTRIBUTE declarations

```
<!ATTLIST poem
          id     ID                               #IMPLIED
          status (draft|revised|published) draft   >
```

In document: `<poem id=P1 status=revised> ... </poem>`

Some possible declared values:

- **CDATA**: any valid character data;
- **SDATA**: system dependent data;
- **NUMBER**: composed only of numerals;
- **ID**: a unique identifier, composed of any valid character data
- **IDREF**: a pointer to an ID

Some possible keywords:

- **#REQUIRED**: a value must be specified;
- **#IMPLIED**: a value need not be supplied;
- **#CURRENT**: if no value is supplied, the last specified value should be used.

Cross-references with IDs

In the DTD:

```
<!ATTLIST poem     id       ID     #IMPLIED  >

<!ELEMENT poemref  - O    EMPTY>

<!ATTLIST poemref  target   IDREF  #REQUIRED >
```

In the document:

```
    <anthology>
      <poem id="p001"><title>The SICK ROSE</title>
         Text of the poem
      </poem>
      <poem id="p002"><title>The Jabberwocky</title>
         Text of the poem
      </poem>
<!-- more poems go here -->
      <poem><title>Hail to thee, o SGML</title>
         This poem has no identifier
      </poem>
    </anthology>

<!-- and somewhere else in the document: -->

Blake's poem on the sick rose <POEMREF TARGET="p001">
```

SGML Entities

Entities are divided into:

- **General entities**

  used in the document instance and divided into:

  - **parameter literals**
  - **external entity specifications**
  - data text
  - bracketed text

- **Parameter entities**

  - **parameter literals**
  - **external entity specifications**
  - **marked sections**

Parameter literals:

In the DTD:

```
<!ENTITY tei "Text Encoding Initiative">
```

In the document:

```
The work of the &tei; has only just begun.
```

After processing:

```
The work of the Text Encoding Initiative has
only just begun.
```

Characters as Entities

Parameter literals are very important for system independent character set representations:

In the DTD:

```
<!ENTITY abreve SDATA "[abreve]"--=small a, breve-->
<!ENTITY Abreve SDATA "[Abreve]"--=capital A, breve-->
<!ENTITY amacr  SDATA "[amacr ]"--=small a, macron-->
<!ENTITY Amacr  SDATA "[Amacr ]"--=capital A, macron-->
<!ENTITY aogon  SDATA "[aogon ]"--=small a, ogonek-->
<!ENTITY Aogon  SDATA "[Aogon ]"--=capital A, ogonek-->
...
```

In the document:

```
<s>&Icirc;ntr-o zi senin&abreve; &scedil;i
friguroas&abreve; de aprilie, pe c&acirc;nd ceasurile
...
```

If the system does not support the character, it can be displayed sloppily:

```
<!ENTITY abreve SDATA "a"--=small a, breve-->
```

If the system does support the character, but it cannot be entered directly, the replacement text is a **character entity**, e.g.:

```
<!ENTITY abreve SDATA "&#123;"--=small a, breve-->
```

External entity specifications

On processing, these entities expand to the contents of the files.

In the DTD:

```
<!ENTITY ChapTwo SYSTEM "sgmlmkup.txt">
```

In the document:

```
<body>
&ChapOne;
&ChapTwo;
...
</body>
```

External entity specifications can refer to `SYSTEM` or `PUBLIC` identifiers.

```
<!ENTITY mte0-ro PUBLIC "-//MTE//TEXT CES1 1984//RO">
```

If `FORMAL=YES` then `PUBLIC` identifier must be properly structured.

Parameter Entities

Used in the **DTD**.

Including entities for the **character set**:

```
<!ENTITY % ISOlat1 PUBLIC
         "ISO 8879-1986//ENTITIES Added Latin 1//EN">
%ISOlat1;
```

Compact **repetitive definitions**:

```
<!ENTITY % a.global '
           id        ID        #IMPLIED
           n         CDATA     #IMPLIED
           lang      IDREF     #IMPLIED' >

<!ATTLIST cesCorpus  %a.global;
          type       CDATA     #IMPLIED
          version    CDATA     #REQUIRED >
```

Marked Sections

In the DTD:

```
<!ENTITY % ENGLISH  "IGNORE">
<!ENTITY % ROMANIAN "INCLUDE">
<!ENTITY % SLOVENE  "IGNORE">
```

In the document:

```
<![ %ENGLISH [
<s>The hallway smelt of boiled cabbage and
old rag mats.</s>
]]>

<![ %ROMANIAN [
<s>Holul blocului mirosea a varz&abreve;
c&abreve;lit&abreve; &scedil;i a pre&scedil;uri
vechi.</s>
]]>

<![ %SLOVENE [
<s>Ve&zcaron;a je smrdela po kuhanem zelju
in starih, cunjastih predpra&zcaron;nikih.</s>
]]>
```

Modularisation in TEI

The **Chicago Pizza** model

In the TEI DTD:

```
<![ %TEI.dictionaries [
<!ENTITY % TEI.dictionaries.dtd PUBLIC
 "-//Text Encoding Initiative//DTD
  P3 12.1: Base Tag Set for Printed Dictionaries//EN//"
>
%TEI.dictionaries.dtd;
]]>
```

When using it:

```
   <!ENTITY % TEI.dictionaries  "INCLUDE">
```

Putting it all together

- every document must refer to a DTD
- every document must contain a DOCTYPE declaration

A complete document:

```
<!DOCTYPE anthology [
    <!ELEMENT anthology   - -  (poem+)>
    <!ELEMENT poem        - -  (title?, stanza+)>
    <!ELEMENT title       - O  (#PCDATA) >
    <!ELEMENT stanza      - O  (line+)   >
    <!ELEMENT line        - -  (#PCDATA) >
]>
    <anthology>
      <poem><title>The SICK ROSE</title>
         <stanza>
              <line>O Rose thou art sick.</line>
              <line>The invisible worm,</line>
              <line>That flies in the night</line>
              <line>In the howling storm:</line>
         <stanza>
              <line>Has found out thy bed</line>
              <line>Of crimson joy:</line>
              <line>And his dark secret love</line>
              <line>Does thy life destroy.</line>
      </poem>
    </anthology>
```

External DTDs

Usually a DTD is external to the document:

```
<!DOCTYPE anthology SYSTEM "antology.dtd">

<!DOCTYPE anthology SYSTEM "antology.dtd" []>

<!DOCTYPE cesCorpus PUBLIC "-//CES//DTD cesDoc//EN">

<!DOCTYPE cesCorpus PUBLIC "-//CES//DTD cesDoc//EN"
[
  <!ENTITY mte1984-en PUBLIC
           "-//MTE//TEXT CES1 1984//EN">
...
  <!ENTITY mte1984-ro PUBLIC
           "-//MTE//TEXT CES1 1984//RO">
  <!ENTITY mte1984-sl PUBLIC
           "-//MTE//TEXT CES1 1984//SL">
]>
<cesCorpus id="o" version="0.1">
  <cesHeader> ... </cesHeader>
  &mte1984-en;
...
  &mte1984-ro;
  &mte1984-sl;
</cesCorpus>
```

Public identifiers & Catalog

(SGML Open Technical Resolution TR9401:1995)

PUBLIC identifiers:

```
"ISO 8879-1986//ENTITIES Added Latin 1//EN"

"-//Text Encoding Initiative//DTD
  P3 3.6.1: Main TEI document type declaration//EN//"

"-//CES//DTD cesDoc//EN"

"-//MTE//ENTITIES Romanian//EN"
```

Mapping to system identifiers (filenames) is performed in the **catalog** file, e.g. for HTML:

```
-- $Id: HTML3.2.catalog,v 1.1
        1996/05/06 connolly Exp $ --

PUBLIC "-//W3C//DTD HTML 3.2 Draft//EN"
       /usr/local/sgml/HTML/HTML32.dtd
PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"
       /usr/local/sgml/HTML/HTML32.dtd
PUBLIC "-//W3C//DTD HTML 3.2//EN"
       /usr/local/sgml/HTML/HTML32.dtd
```

The complete picture

- the **SGML declaration**,
  if not using the RCS
- the **catalog file(s)**,
  if using public identifiers
- the **DTDs**,
  stored as specified in the catalog or SYSTEM identifiers
- the **document instances**,
  that you are creating or using
- **SGML aware software**,
  to do something with all this

Processing SGML

No SW implements the complete SGML standard. But some come close.

SGML aware SW is either free or very expensive.

SGML software components:

- editing tools and environments
- conversion engines and applications
- the document management system
- formatting engines for rendering on paper and screen
- other transformation engines
- search and retrieval engines
- additional processing applications
- the "document engineering toolbox"

SGML validating parser

The basic question for a developer:

- Is the document well formed?
- Getting a normalised instance of the document

The best choice:

James Clark's `SP`

SP is a "object-oriented toolkit for SGML parsing and entity management."

- implements most of the standard
- freely available
- runs on most platforms
- bug-free

Validating with nsgmls

The document `mixed.sgml`:

```
<!doctype listitem [
   <!ELEMENT listitem  - - (#PCDATA, para+)>
   <!ELEMENT para      - - (#PCDATA)>
]>
<listitem>
Oranges
<para>Oranges are a lovely fruit.</para>
<para>They grow in warm climates.</para>
</listitem>
```

`nsgmls` as validator:

```
nsgmls -s mixed.sgml
nsgmls:mixed.sgml:7:40:E: character data is not allowed here
nsgmls:mixed.sgml:8:40:E: character data is not allowed here

SGML validation exited abnormally with code 1
```

Parsing with `nsgmls`

Fixed document:

```
<!doctype listitem [
   <!ELEMENT listitem  - - (#PCDATA, para+)>
   <!ELEMENT para      - - (#PCDATA)>
]>
<listitem>
Oranges
<para>
Oranges are a lovely fruit.
</para><para>
They grow in warm climates.
</para></listitem>
```

`nsgmls` ESIS output:

```
nsgmls mixed.sgml
(LISTITEM
-Oranges\n
(PARA
-Oranges are a lovely fruit.
)PARA
(PARA
-They grow in warm climates.
)PARA
)LISTITEM
C

SGML validation finished at Fri Jul 11 13:26:51
```

Editing

Freely available: **(X)Emacs** & **psgml mode**