Overview of the FreeBSD Documentation Framework

Hiroki Sato <hrs@FreeBSD.org>
FreeBSD Documentation Project

2011/10/6 @ EuroBSDCon 2011 DevSummit

Agenda

- Directory Structure
- Documents and Processing Flow
  - The www/ directory and HTML docs
  - The doc/ directory and SGML docs
  - XML
- Discussions and Future Directions

Directory Structure

- The www/ and doc/ directories in the CVS repo.
- www = contents for www.FreeBSD.org
- doc = collection of FreeBSD documents in more structured format (i.e. SGML)

Why Separated 2 dirs?

- Designed by nik@FreeBSD.org back in 1996
  - Many documents in plain text and HTML.
    - Plain text and HTML documents were converted to SGML with LinuxDoc DTD first.
    - LinuxDoc documents were converted to DocBook DTD.
- Motivation = Maintaining a lengthy document in plain text or HTML was hard.

Why Separated 2 dirs?

- Problems when adopting SGML were:
  - SGML was promising and better than plain HTML.
  - But, learning SGML was pain for developers.
  - People tended to avoid writing documents.

Why Separated 2 dirs?

- At that time even HTML was blamed: "why plain text is not enough?"
- A compromise suggested by nik@ was:
  - www = plain HTML
  - doc = DocBook SGML
- HTML was in the middle of 3.2 and 4.0.

The www/ Directory

- Use the HTML format as an SGML application.
  - HTML is defined by using the SGML specification.
  - An HTML document is also an SGML document.
- Documents in the www dir are written in "SGML".

The www/ Directory

- Differences between HTML and SGML
  - In an SGML doc, entity reference like © or &#20;. can be defined. HTML has a pre-defined entity set only.
  - SGML can define character sets and encodings. HTML can specify some pre-defined encodings only.
  - SGML can define new elements and attributes like <foo attr=”bar”> using DTD or schema. HTML cannot.

The www/ Directory

- What we are doing for documents in the www/ dir are:
  - Validate the documents using HTML DTD.
  - Normalize the documents to make it ready for www browsers. Case in element tags, expansion of &foo; reference, etc.

www/en/docs.sgml

```
<!DOCTYPE HTML PUBLIC "-//FreeBSD//DTD HTML 4.01 Transitional-Based
Extension//EN" [
<!ENTITY date "$FreeBSD: www/en/docs.sgml,v 1.195 2006/08/19 21:20:30
hrs Exp $">
<!ENTITY title "FreeBSD Documentation">
<!ENTITY % navinclude.docs "INCLUDE">
]>

<html>
  &header;

  <img src="gifs/doc.jpg" alt="BSD Daemon reading documentation"
    align="right" border="0" width="274" height="163">

  <p>A wide variety of documentation is available for FreeBSD,
    on this web site, on other web sites, and available over
    the counter.</p>

    &footer;
 </body>
</html>
```

www/en/docs.sgml

preamble

```
<!DOCTYPE HTML PUBLIC "-//FreeBSD//DTD HTML 4.01 Transitional-Based
Extension//EN" [
<!ENTITY date "$FreeBSD: www/en/docs.sgml,v 1.195 2006/08/19 21:20:30
hrs Exp $">
<!ENTITY title "FreeBSD Documentation">
<!ENTITY % navinclude.docs "INCLUDE">
]>
```

body

```
<html>
  &header;

  <img src="gifs/doc.jpg" alt="BSD Daemon reading documentation"
    align="right" border="0" width="274" height="163">

  <p>A wide variety of documentation is available for FreeBSD,
    on this web site, on other web sites, and available over
    the counter.</p>

  &footer;
 </body>
</html>
```

www/en/docs.sgml

<!DOCTYPE HTML PUBLIC "-//FreeBSD//DTD HTML 4.01 Transitional-Based Extension//EN" [
<!ENTITY date "$FreeBSD: www/en/docs.sgml,v 1.195 2006/08/19 21:20:30 hrs Exp $">
<!ENTITY title "FreeBSD Documentation">
<!ENTITY % navinclude.docs "INCLUDE">
]>

- <!DOCTYPE> declaration means start of the SGML doc.
  - "-//FreeBSD//..." is a DTD identifier. PUBLIC means it is a public DTD.
  - <HTML> is the top level element.
  - Contents between "[" and "]" will be added to the head of the DTD.

www/en/docs.sgml

<!DOCTYPE HTML PUBLIC "-//FreeBSD//DTD HTML 4.01 Transitional-Based Extension//EN" [
<!ENTITY date "$FreeBSD: www/en/docs.sgml,v 1.195 2006/08/19 21:20:30 hrs Exp $">
<!ENTITY title "FreeBSD Documentation">
<!ENTITY % navinclude.docs "INCLUDE">
]>

- The DTD identifier must be defined in share/sgml/catalog. SGML toolchain looks for the substance by using it.

--
 $FreeBSD: www/share/sgml/catalog,v 1.1 2006/08/19 21:20:53 hrs Exp $
--

PUBLIC "-//FreeBSD//DTD HTML 4.01 Transitional-Based Extension//EN"
       "html401-freebsd.dtd"

www/en/docs.sgml

<!DOCTYPE HTML PUBLIC "-//FreeBSD//DTD HTML 4.01 Transitional-Based Extension//EN" [
<!ENTITY date "$FreeBSD: www/en/docs.sgml,v 1.195 2006/08/19 21:20:30 hrs Exp $">
<!ENTITY title "FreeBSD Documentation">
<!ENTITY % navinclude.docs "INCLUDE">
]>

- DTD can include another DTD by using <!ENTITY>. Another DTD is referred by the DTD identifier, and it involves catalog look-up (“catalog chain”)

!-- $FreeBSD: www/share/sgml/html401-freebsd.dtd,v 1.1 2006/08/19 21:20:53 hrs Exp $ -->
  :
<!ENTITY % l10n.ent PUBLIC "-//FreeBSD//ENTITIES FreeBSD L10N Entities//EN">
%l10n.ent;
  :
<!ENTITY % html.orig PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
%html.orig;

<!ENTITY>

```
<!DOCTYPE HTML PUBLIC "-//FreeBSD//DTD HTML 4.01 Transitional-Based
Extension//EN" [
<!ENTITY date "$FreeBSD: www/en/docs.sgml,v 1.195 2006/08/19 21:20:30
hrs Exp $">
<!ENTITY title "FreeBSD Documentation">
<!ENTITY % navinclude.docs "INCLUDE">
]>
```

- <!ENTITY> defines entity which can be referred by &foo; or %foo;.
- "%" is special and only used within DTD.

```
<!ENTITY title "FreeBSD Documentation">
 &title;

<!ENTITY % navinclude.docs "INCLUDE">
 %naviinclude.docs;
```

<!ENTITY>

```
<!DOCTYPE HTML PUBLIC "-//FreeBSD//DTD HTML 4.01 Transitional-Based
Extension//EN" [
<!ENTITY date "$FreeBSD: www/en/docs.sgml,v 1.195 2006/08/19 21:20:30
hrs Exp $">
<!ENTITY title "FreeBSD Documentation">
<!ENTITY % navinclude.docs "INCLUDE">
]>
```

- <!ENTITY> can also be used for including a file.

```
<!ENTITY file SYSTEM "file.ent">
 &file;
```

- Using PUBLIC identifier + catalog, not SYSTEM, is preferred, though.

<!ENTITY>

<!DOCTYPE HTML PUBLIC "-//FreeBSD//DTD HTML 4.01 Transitional-Based Extension//EN" [
<!ENTITY date "$FreeBSD: www/en/docs.sgml,v 1.195 2006/08/19 21:20:30 hrs Exp $">
<!ENTITY title "FreeBSD Documentation">
<!ENTITY % navinclude.docs "INCLUDE">
]>

- What if two <!ENTITY> conflict each other?
  - The first one wins.
  - Inclusion order of DTDs are very important.
- The above example, "[...]" will come BEFORE the "-//FreeBSD//..." DTD.

www/en/docs.sgml

<!DOCTYPE HTML PUBLIC "-//FreeBSD//DTD HTML 4.01 Transitional-Based Extension//EN" [
<!ENTITY date "$FreeBSD: www/en/docs.sgml,v 1.195 2006/08/19 21:20:30 hrs Exp $">
<!ENTITY title "FreeBSD Documentation">
<!ENTITY % navinclude.docs "INCLUDE">
]>

- Three ENTITYs, &date;, &title;, %navinclude.docs;, are defined here, and they will NEVER be overridden.
- Other definitions such as <p> element are included from HTML 4.01 DTD. (You do not want to read it because it is quite complex.)

www/en/docs.sgml

preamble

```
<!DOCTYPE HTML PUBLIC "-//FreeBSD//DTD HTML 4.01 Transitional-Based
Extension//EN" [
<!ENTITY date "$FreeBSD: www/en/docs.sgml,v 1.195 2006/08/19 21:20:30
hrs Exp $">
<!ENTITY title "FreeBSD Documentation">
<!ENTITY % navinclude.docs "INCLUDE">
]>
```

body

```
<html>
  &header;

  <img src="gifs/doc.jpg" alt="BSD Daemon reading documentation"
    align="right" border="0" width="274" height="163">

  <p>A wide variety of documentation is available for FreeBSD,
    on this web site, on other web sites, and available over
    the counter.</p>

  &footer;
 </body>
</html>
```

www/en/docs.sgml

```
<html>
→ &header;

  <img src="gifs/doc.jpg" alt="BSD Daemon reading documentation"
    align="right" border="0" width="274" height="163">

  <p>A wide variety of documentation is available for FreeBSD,
    on this web site, on other web sites, and available over
    the counter.</p>

→   &footer;
  </body>
</html>
```

- header/footer templates are defined as &header; and &footer;. The &header; includes <body>.
- A document writer can focus on writing contents in HTML between the two entity references. No special SGML knowledge is needed.

SGML toolchain for www

- SGML toolchain used in the www directory
  - sgmlnorm: SGML validator and normalizer. A tool in a package "SP" by James Clark (a famous person in text processing)
  - tidy: indent(1)-like tool for HTML

```
% make docs.html
/usr/bin/sed -e 's/<!ENTITY date[ \t]*"$Free[B]SD. .* \(.* .*\) .* .* $">/<!
ENTITY date "Last modified: \1">/'   docs.sgml > docs.sgml-tmp

/usr/bin/env SGML_CATALOG_FILES=  /usr/local/bin/sgmlnorm -d -
ifreebsd.urls.absolute -c /usr/local/share/sgml/html/catalog -c /usr/local/share/
sgml/catalog -c /home/hrs/freefall/www/en/share/sgml/catalog -c /home/hrs/
freefall/www/share/sgml/catalog -D /home/hrs/freefall/www/en docs.sgml-tmp
> docs.html ||  (/bin/rm -f docs.sgml-tmp docs.html && false)

/bin/rm -f docs.sgml-tmp
/usr/local/bin/tidy -wrap 90 -m -raw -preserve -f /dev/null -asxml  docs.html
*** Error code 1 (ignored)
```

SGML toolchain for www

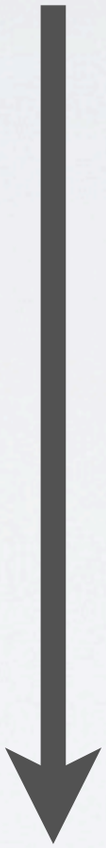
- References
  - $LOCALBASE/share/doc/jade/*.htm
    - "SP" is installed as a part of textproc/jade with no manual page.
  - textproc/opensp is a newer version of SP. No longer actively maintained, unfortunately.

The doc/ Directory

- Container of SGML documents
  - Articles - collection of short documents about specific topics.
  - Books - collection of lengthy documents
  - FAQ - collection of Q&A.

Building Docs in doc/

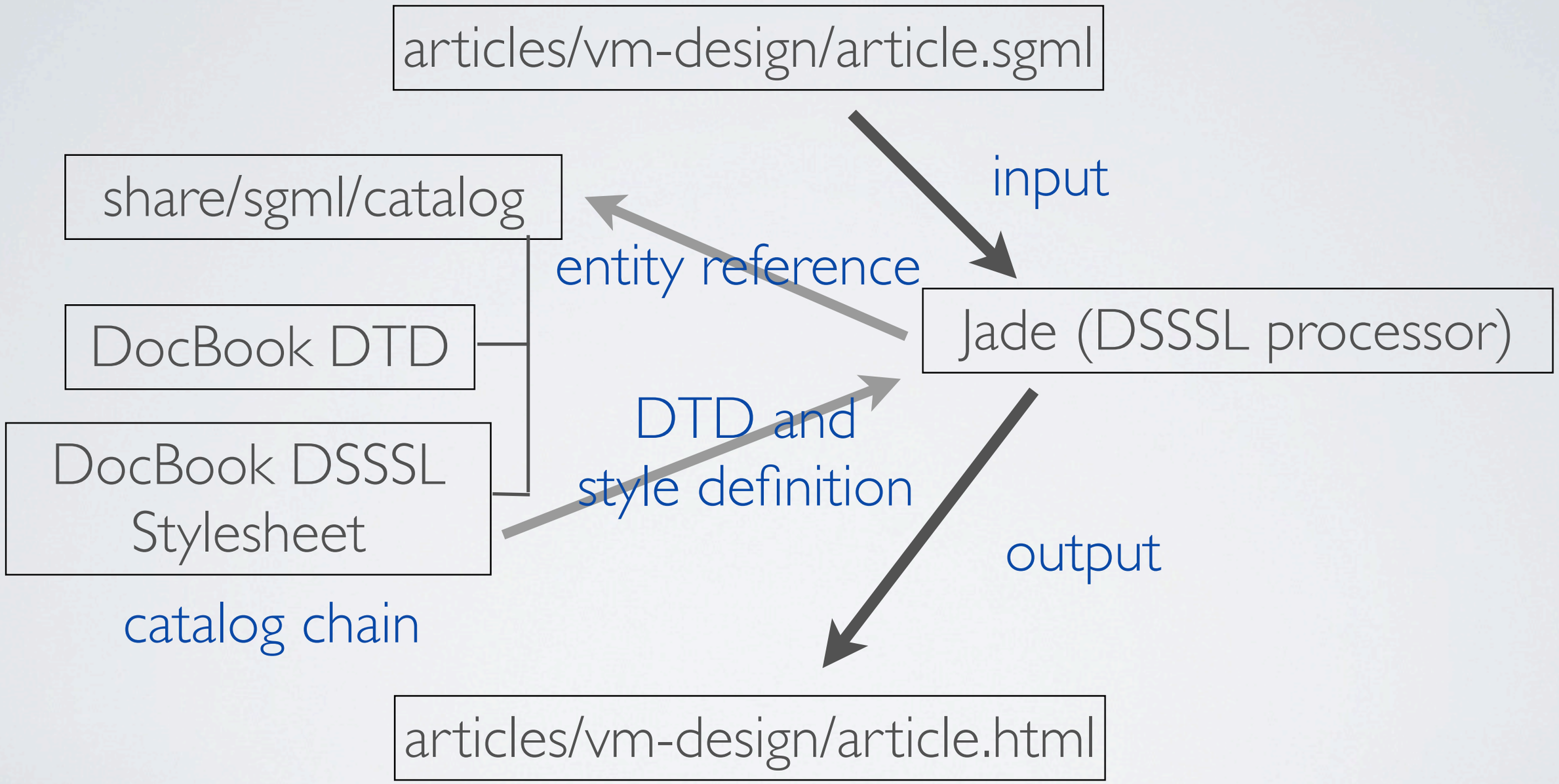
articles/vm-design/article.sgml



SGML to HTML

articles/vm-design/article.html

Building Docs in doc/



DSSSL and Jade

- Document Style Semantics and Specification Language (DSSSL), pronounced as "dee-sl".
- Defined in ISO/IEC 10179:1996.
  - Scheme (a LISP variant) based programming language to specify "style" of an SGML document. Consists of Transformation language (querying a node) and Style language (applying a style).
  - "Jade" is an implementation of DSSSL processor by James Clark.
  - AFAIK, in the world there are only two practical ones which we can get easily.

A Simple Example of DSSSL

DSSSL

```
(default
   (process-children))

(element para
   (make element gi:"P"
            attributes:'(("CLASS" "paragraph"))
      (process-children)))
```

Input SGML

```
<article>
     <para>foo
          <para>bar</para>
     </para>
</article>
```

Output

```
<p CLASS="paragraph>foo<p CLASS="paragraph">bar</p></p>
```

A Simple Example of DSSSL

DSSSL

```
(default
  (process-children))

(element para
  (make element gi:"P"
        attributes:'(("CLASS" "paragraph"))
    (process-children)))
```

Define style rules for each element

Input SGML

```
<article>
    <para>foo
        <para>bar</para>
    </para>
</article>
```

Target

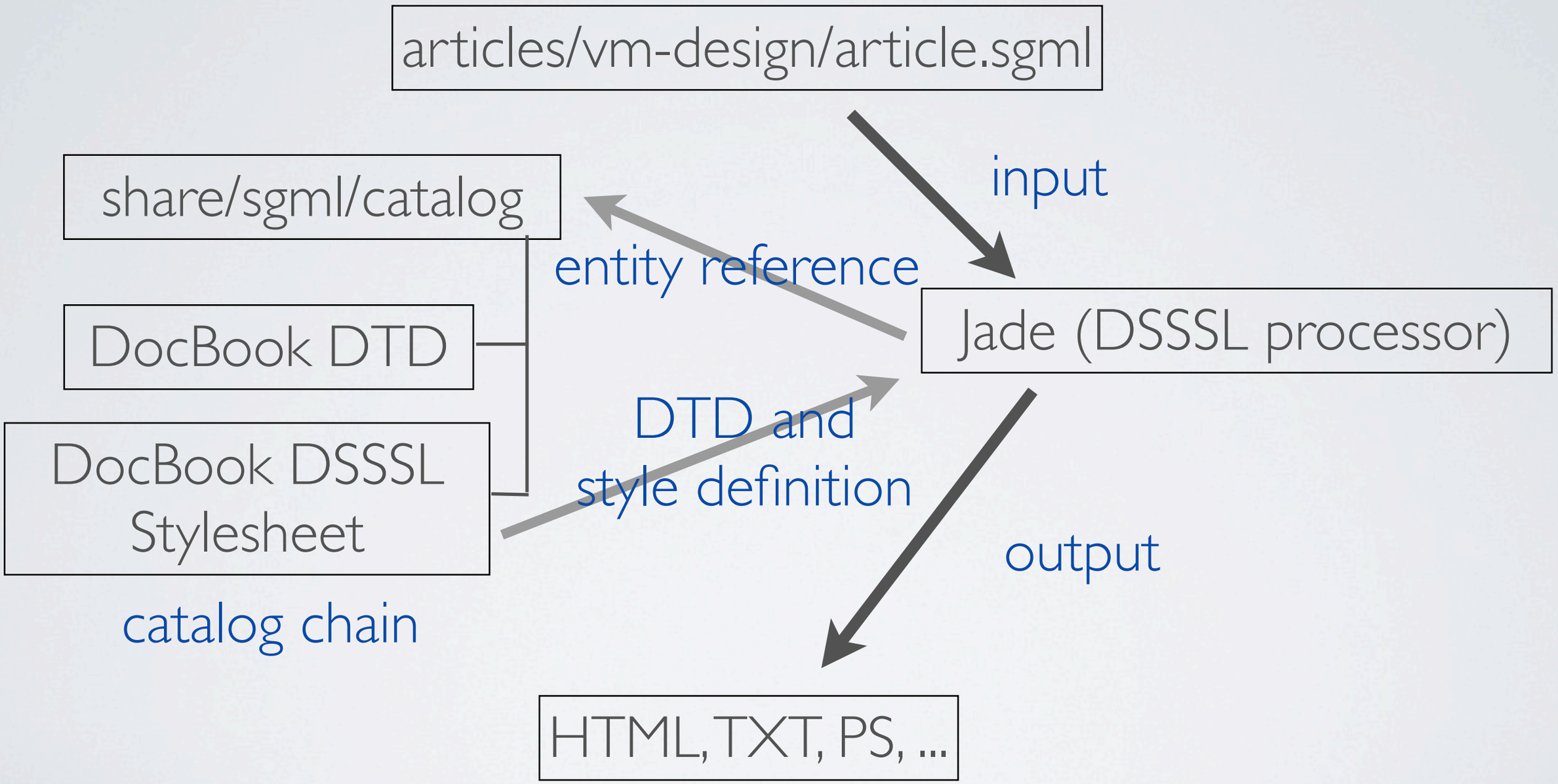
Output

```
<p CLASS="paragraph>foo<p CLASS="paragraph">bar</p></p>
```

Output Formats by Jade

- DSSSL can generate various output formats:
  - SGML (another DTD), Plain text, PostScript, ...
    - DSSSL works as a generic converter from SGML to other formats.
  - PS and PDF:
    SGML + DSSSL → JadeTeX → PS or PDF

Building Docs in doc/



XML

- The successor of SGML.
- Difference between XML and SGML:
  - UTF-8 is used internally.
  - Entity reference is slightly changed.
  - Close-tag omission is no longer supported.
  - Namespace, case-sensitivity.
- Conversion from SGML to XML is relatively easy.
  sx(1) in SP package will do this.

XSLT

- The successor of DSSSL by W3C.
- XML Stylesheet Language Transformations
  - libxslt fully supports XSLT 1.0. XSLT 1.2 was not standardized, XSLT 2.0 is now the current version.
  - xsltproc(1), a C/C++ implementation.

XSLT

DSSSL

```
(element para
   (make element gi:"P"
         attributes:'(("CLASS" "paragraph"))
      (process-children)))
```

XSLT

```
<xsl:template match="para">
   <xsl:element name="P">
      <xsl:attribute name="CLASS">
         <xsl:value-of select="paragraph" />
      </xsl:attribute>
      <xsl:value-of select="." />
   </xsl:element>
</xsl:template>
```

XSLT

DSSSL

```
(element para
  (make element gi:"P"
        attributes:'(("CLASS" "paragraph"))
    (process-children)))
```

XSLT

```
<xsl:template match="para">
  <xsl:element name="P">
    <xsl:attribute name="CLASS">
      <xsl:value-of select="paragraph" />
    </xsl:attribute>
    <xsl:value-of select="." />
  </xsl:element>
</xsl:template>
```

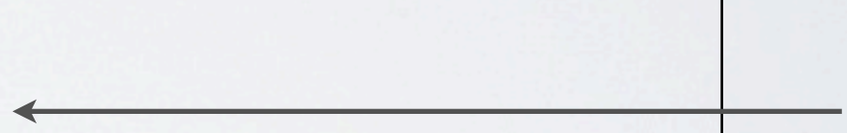
XSLT

DSSSL

```
(element para
  (make element gi: "P"
        attributes: '(("CLASS" "paragraph"))
    (process-children)))
```

XSLT

```
<xsl:template match="para">
  <xsl:element name="P">
    <xsl:attribute name="CLASS">
      <xsl:value-of select="paragraph" />
    </xsl:attribute>
    <xsl:value-of select="." />
  </xsl:element>
</xsl:template>
```



XSLT

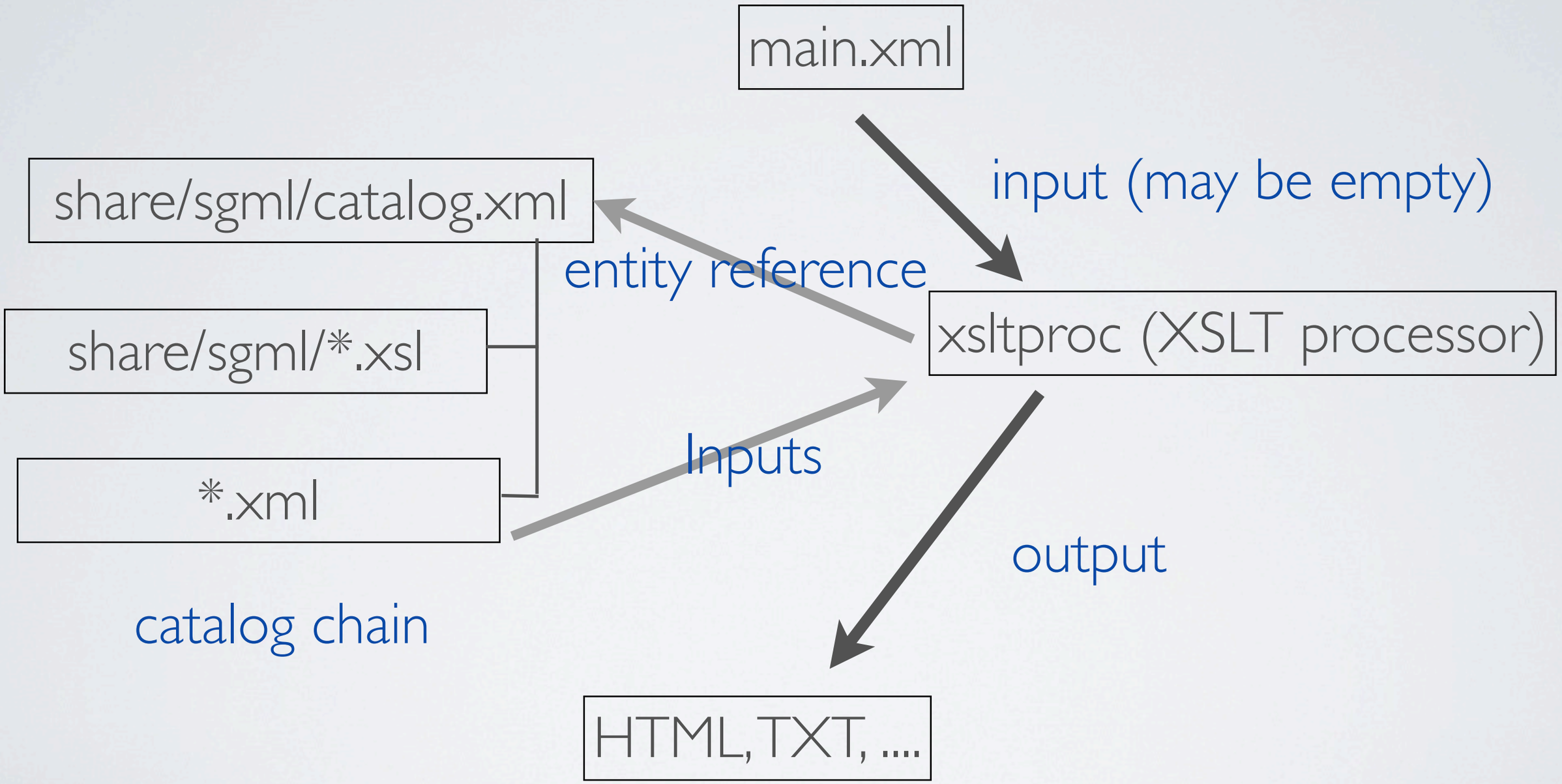
DSSSL

```
(element para
   (make element gi:"P"
         attributes:'(("CLASS" "paragraph"))
      (process-children)))
```

XSLT

```
<xsl:template match="para">
   <xsl:element name="P">
      <xsl:attribute name="CLASS">
         <xsl:value-of select="paragraph" />
      </xsl:attribute>
      <xsl:value-of select="." />
   </xsl:element>
</xsl:template>
```

XML + XSLT Processing Flow



XSLT 1.0, 1.2, 2.0, ...

- XSLT is (should be?) powerful enough to convert DSSSL stylesheet.
- Actually, it is almost equivalent or superior to DSSSL.
- However, XSLT 1.0 has some design flaws.

  A template can call another template like sub-routine, but the result can be obtained as flattened string, not a node set.

  This often makes using a template as a sub-routine very difficult.

Best Practices of XML/SGML

- Information Reusability:

  XML + XSLT makes possible to store information in one place, and reuse it multiple times in various ways.

- Separation of Style and Structure:

  XML should contain the contents, XSLT should contain the structure and style in the output format. DO NOT mix them.
  Our index.xsl is a bad example, unfortunately.

- Note: these factors make some of us dislike adopting wiki throughout our documentation set. Use the right tool for the right purpose.

Best Practices of XML/SGML

- Information Reusability:

  XML + XSLT makes possible to store information in one place, and reuse it multiple times in various ways.

  Multiple processing possible:
  (((XML + XSLT) + XSLT) + ...) = XML

- Separation of Style and Structure:

  XML should contain the contents, XSLT should contain the structure and style in the output format. DO NOT mix them. Our index.xsl is a bad example, unfortunately.

Best Practices of XML/SGML

- Separations for the information reusability:

  Contents/Structure separation
  - having index.xml + index.xsl
  - translated where.sgml can be broken easily

  LI/LD separation (translation)
  - doc/$LANG/share/sgml v.s. doc/share/sgml
  - Consider English is one of our supported languages.

Future Direction and Discussions

- CVS to SVN conversion
  doceng@ is working on this but it is behind (sorry!)
  A prototype of the conversion is ready, and will be released within this month.

- similar structure to src:
  - head/doc/$LANG/articles
    head/doc/$LANG/books
    head/doc/$LANG/htdocs ← www/$LANG
    head/doc/$LANG/share/sgml ← www/$LANG/share/sgml
    head/doc/share/sgml ← www/share/sgml

  - release/9.0/doc for each releases

  - user/ and project/ for committers

Future Direction and Discussions

- SGML to XML conversion
  - DocBook/SGML 4.2-based SGML files should be converted to DocBook/XML 5.X.
  - DSSSL to XSLT is still a bit pain because of XSLT 1.0's flaw.
    - XML and XSLT toolchains. XML, XSLT, XInclude, XPath, ...
      xsltproc works well for XML-XML conversion.
      xmlroff for XML-FO (formatting object) or PDF conversion.
    - Jade + XML? Partially possible. Jade knows very initial draft specification of XML.
- When: after CVS-SVN conversion. Create a working branch.

Future Direction and Discussions

- Some non-static pages
  - Portal pages for per-port, per-subsystem, per-device driver, etc. Gather all of information about a specific part like PR, commits, contacts, forums, wiki pages into one page.
- hardware compatibility list (recently popped up again in -doc@). Maintain database by users.