

DSSSL

DSSSL

- Document Style Semantics and Specification Language
- Used to Specify style sheets for SGML documents

Why use DSSSL?

- SGML is machine readable
- SGML is not very human readable
- DSSSL serves to convert data in SGML into more readable form
- Some format it can convert to are RTF, HTML, Latex

Major Components of DSSSL

- Style Language
- Flow Object
- Transformational Language
- Document Model
- Query Language

Style Language

- Describes the formatting of SGML (XML) documents
- Based heavily on Scheme (both syntax and functionality)
- A "grove" is constructed from the SGML document which is formatted into a flow object by style rules.
- A processor then converts the tree to a document

Style Language Examples

```
(element p  
  (make paragraph font-size: 12pt))
```

```
(element (ul li)  
  (make paragraph ...))
```

```
(element empty  
  (empty-sosofa))
```

```
(define four (+ 2 2))
```

Flow Objects

- Provide a mechanism for describing the layout of a document
- Pages, paragraphs, tables, image, etc
- Formatting specifications are contained within the construction rules

```
(make paragraph
  font-size: 12pt
  font-family-name: "Arial"
  ....)
```

```
(make sequence
  font-posture: 'italic
  ...)
```

Transformation Language

- Superset of the Expression language
- Used to transform one SGML document to another
- Allows for the creation and manipulation of document nodes
- Not widely used (and Jade doesn't even implement it)

Document Model

- The Tree that gets generated from an SGML document
- The entire tree is known as a "grove" in the DSSSL world
- Elements, attributes, etc are nodes in the tree

Query Language

- Used to select individual nodes from the grove
- Functions to access attributes, children, ancestors, etc
- Functions to retrieve a nodes position in the tree

DSSSL vs XSL

DSSSL	XSL
Style Language	XSLT
Flow Object	XSL-FO
Transformation Language	XSLT
Document Model	XML Information Set
Query Language	XPath

Construction Rules

- DSSSL Documents consist mostly of construction rules
- Map elements in the source document to DSSSL flow objects
- Example: the para tag creates a new paragraph with a 12pt font

```
(element para
  (make paragraph
    font-size: 12pt))
```

Process Children

- (process-children) returns the tree that results from processing the children of a node
- The results are all linked to the root node
- Similar to "apply-templates" in XSLT

Controlling Recursion

- Make a new paragraph containing the processed children

```
(element para (make paragraph))
```

- Make a paragraph containing "foo" and ignores para's children

```
(element para (make paragraph (literal "foo")))
```

- Make a paragraph containing the contents of the first title tag under the para

```
(element para  
  (make paragraph  
    (process-first-descendent "title")))
```

Modes

- Often you want different rules to apply to the same element depending on the context
- Table of contents for example
- Modes allow the specification of construction rules that only sometimes apply

```
(mode toc
  (element chapter ...)
  (element section ...)
  (element para (empty-sosof)))
(element book
  (make sequence
    (with-mode toc (process-children))
    (process-children)))
```

Example: HTML Subset

```
(element html (make simple-page-sequence))
```

```
(element h1  
  (make-paragraph  
    font-weight: 'bold  
    font-size: 20pt  
    space-after: 10pt  
    quadding: 'center))
```

```
(element p  
  (make-paragraph  
    font-size: 12pt  
    quadding: 'start))
```

```
(element i  
  (make sequence  
    font-posture: 'italic))
```

Example: This presentation

- This presentation is generated from an XML document by DSSSL
- This slide looks something like

```
<slide type="normal">
  <title>Example: This presentation</title>
  <body>
    <item>This presentation is generated from
      an XML document by DSSSL</item>
    <item>This slide looks something like</item>
    <code>
<lt;slide type="normal"> ...
  </code>
  </body>
</slide>
```

Questions?