Outline

Introduction 21

1 **XML and the Web** 1

2 **The source definition** 39

3 **Elements of a web site** 91

4 **An overview of XSLT** 155

5 **The XSLT stylesheet** 183

6 **XML software** 277

7 **XML on the server** 355

Bibliography 385

Index 387

Contents

Introduction xxi

chapter 1 *XML and the Web* 1

1.1 **Content, presentation, structure** 4

Computer as a language 4

1.1.1 The stairway of abstractions 5

1.1.2 Document oppositions 6

Convertibility 6 *Style separation* 7 *Richness of structure* 7
Modularity 8 *Size* 8

1.1.3 The role of XML 8

Human-readable 9 *Arbitrarily rich* 9 *Rather bulky* 9

1.1.4 The role of HTML 9

What you can and cannot do with HTML 9 *Other target formats* 10
Visualizing XML 11

Figure 1.1 11
Figure 1.2 12

1.1.5 The Semantic Web 11

The Extensible Meaning Language 13

1.2 **The two faces of XML** 13

Data is regular 14 *Data is shallow* 14 *Documents use mixed content* 14
Documents are ordered 15 *Example analysis* 15

1.3 **Components of an XML web site 15**

1.3.1 Source documents **16**

In your own tongue **16** *Master and slaves* **16**

1.3.2 Validation **16**

Schematron rules **17**

1.3.3 Transformation stylesheet **17**

Variables, parameters, and functions **17** *Trunk templates* **17** *Branch templates* **17** *Extension Java classes* **18** *Batch processing* **18**

1.3.4 Static objects **18**

1.3.5 Integration **19**

1.4 **Setting up an XML web site 19**

1.4.1 XML offline **20**

Works with existing servers and clients **20** *Server performance is optimal* **20** *Use any software for XML processing* **20** *Not for dynamic sites* **21** *Offline updates may be slow* **21** *Only author can access XML* **22** Figure 1.3 **21**

1.4.2 XML on the server **22**

Dynamic sites are OK **22** *No special clients required* **22** *Set up your server as necessary* **23** *Everyone can access XML* **24** *Server performance may suffer* **24** *Server setup may be complicated* **24** *Separate testing of pages is necessary* **24** Figure 1.4 **23**

1.4.3 XML in the browser **25**

Power to the users **25** *XML is accessible from the server* **26** *XML support in the client is required* **27** *Server-side transformation is still necessary* **27** *XSLT extensibility is limited* **27** *Viewing performance may suffer* **27** *And what about us?* **28** Figure 1.5 **26**

1.5 **XML and dynamic sites 28**

1.5.1 The two sources **29**

1.5.1.1 Dynamic values **29**

Self-typing atoms **29** *Mixed-content composites* **29**

1.5.1.2 Static templates **30**

Figure 1.6 **31**
Figure 1.7 **32**

1.5.1.3 Do you need XML? **30**

1.5.2 The two scenarios **33**

1.5.2.1 Compile, then transform **33**

Before this is possible **35** *"Compile, then transform" scorecard* **35** *Compile and transform offline* **36** Figure 1.8 **34**

1.5.2.2 Transform, then compile **37**

Figure 1.9 **38**

chapter 2 *The source definition* 39

2.1 **The big picture** 42

Defining the definition 42 *Vocabulary alert* 42 *The ways of being correct* 42 *After design but before implementation* 43

2.1.1 Two-tier architecture 43

Document layer 43 *Super-document layer* 44 *Information distribution* 44 *File conventions* 44 *File-to-file correspondences* 44 *File-to-element correspondences* 44

2.1.1.1 Implementing the super-document rules 45

Human-readable instructions 45 *Back-end scripting* 45 *Build layer checks* 46 *Stylesheet checks* 46 *Schematron schemas* 47 *Greet the winner* 48 *Do not forget to super-document* 48

2.1.2 Organizing source documents 48

2.1.2.1 Master document 49

Find it on the map 49 *Site directory* 49 *What it is not* 49 Figure 2.1 50

2.1.2.2 Orthogonal content 51

Examples 51 *Referencing orthogonals* 51 *Not where it appears to be* 52 *External entities considered harmful* 53

2.1.2.3 Storing auxiliary data in the stylesheet 54

Boil-down analysis 54

2.1.2.4 Other approaches 55

Everything in one chunk 55 *Objects in a database* 55 *Reusing existing infrastructure* 55

2.2 **Practical schematization** 55

2.2.1 Choosing the language 56

2.2.1.1 Languages for building grammars 56

Grammar limitations 57

2.2.1.2 Languages for setting rules 57

Precision aiming 58 *Schematron primer* 58 *Growing rules into grammar* 59 *Guided editing* 60 *Best of both worlds* 60 Example 2.1 59

2.2.1.3 Modularity 60

XSDL 61 *DTDs* 61 *Schematron* 62

2.2.1.4 Expressiveness 63

Fake integers in DTDs 63 *This is a complete sentence* 63 *Document the schema* 64 *Provide validation-time diagnostics* 64 *Choose a "talking" name* 64 Example 2.2 65

2.2.1.5 Strictness 65

2.2.2 Schema creation scenarios 66

2.2.2.1 Working incrementally 67

Starting small 67 *Ending big* 68

2.2.2.2 Changing the rules 68
2.2.3 Documenting schemas 69
2.2.3.1 Documenting in different languages 69
DTDs 70 *XSDL* 70 *Schematron* 70
2.2.3.2 Documentation components 71
Any relevant rules that are not in the schema itself 71 *Markup examples* 71 *Structural information* 72 *Metadata* 73 Example 2.3 72
2.2.3.3 Page templates 73
2.2.4 Using DTDs 74
2.2.4.1 DTDs and namespaces 74
2.2.4.2 Linking DTDs to documents 75
2.2.4.3 Mnemonic entity references 76
2.2.4.4 External entities 77
2.3 **The art of source definition** 77
2.3.1 Semantic analysis 78
2.3.2 Learn to think hierarchically 78
2.3.3 Child elements vs. attributes 79
Attributes are not extensible 79 *Attributes are unique within their elements* 79 *Attributes are unordered* 79
2.3.4 The art of naming 81
Spell out 81 *Use case* 81 *Hyphen-ate* 81 *Readable markup* 81
2.3.5 Structure vs. metadata 82
Example: translations 83
2.3.6 Generalizing but not overgeneralizing 83
Troublesome heritage 84
2.3.7 Parallel vs. sequential 84
Anatomy of a menu item 84
2.3.8 Existing vocabularies 86
Concepts 86 *Interoperability* 86 *Completeness* 87
2.3.9 Namespace strategies 87

chapter 3 *Elements of a web site* 91

3.1 **Page documents: top-level structures** 94
3.1.1 Page metadata 94
Page ID 94 *Page coordinates* 95 *Everything else* 95 *Existing vocabularies* 95

3.1.2 Sections and blocks 95
Sections or blocks? 95 *Block types* 96

3.2 **Headings** 97

3.2.1 Element type names 97
Look up the number 97 *Ask my parent who I am* 97
3.2.2 Attributes 98
3.2.3 Children 99
3.2.4 Web page title 100
Multistage titles 100

3.3 **Paragraphs** 101

3.3.1 Lists 101
3.3.2 Paragraphs as link targets 102
3.3.3 Displayed material 103

3.4 **Text markup** 103
Block and inline elements 103 *Existing vocabularies* 104
3.4.1 Mark up the meaning 105
3.4.2 Rich markup 105
Existing vocabularies 107
3.4.3 Transcending levels 107
3.4.4 Nested markup 109

3.5 **Links** 109

3.5.1 Elements or attributes? 109
3.5.2 Link types 111
Categorizing links 111 *Classifier attributes* 112 *Classifier element types* 112 *Advanced link types* 112
3.5.3 Abbreviating addresses 113
Example: RFC links 114 *Mnemonic addressing* 114
3.5.3.1 Multiple abbreviation schemes 115
3.5.3.2 Deabbreviation algorithms 115
3.5.3.3 Multicomponent abbreviations 116
3.5.3.4 Internal links 116
Linking a foobar 117 *Aliases* 117 *Linking translations* 117

3.6 **Images and objects** 118
Element type names 119 *Images as attributes* 119
3.6.1 Abbreviating location 119
Abbreviating aggressively 120

3.6.2 Formatting hints 120
Think ahead 121 *Separate namespaces* 121
3.6.3 Image metadata 122
Textual descriptions 122 *Captions* 123
3.6.4 Imagemaps and interactive objects 123
The quick-and-dirty approach 123 *The thoroughly semantic approach* 124 *Accessibility* 124
3.7 **Tables 125**
If you have something you can name, do it 125 *Tables from triplets* 126 *Is it worth it?* 126
3.8 **Forms 126**
Existing vocabularies 127 *Formatting hints* 128
3.9 **Master document 129**
3.9.1 Site structure 130
3.9.1.1 Menu structure 130
3.9.1.2 Menu items and pages 131
Items vs. pages 131 *How deabbreviation works* 132 *Accessing the source* 132 *Storing page metadata* 132
3.9.1.3 Orthogonal content 133
Extracting orthogonal content 133 *No perfection in this world* 135
3.9.1.4 Registering dynamic content 135
One way of many 135 *Reusing blocks* 136 *Calling a process* 136 *Watching a directory* 137 *XML, not HTML* 137
3.9.2 Common content and site metadata 138
3.9.3 Processing parameters 139
Grouping parameters into environments 139 *Where to store the environment groups?* 139
3.9.4 Site-wide content and formatting 140
Site-wide buttons 140
3.10 **Summary examples 141**
3.10.1 Page document 141
Example 3.1 141
3.10.2 Master document 142
Languages 142 *Environments* 143 *Menu* 143 *Blocks* 144 *Misc* 144 Example 3.2 144
3.10.3 Schematron schema 148
Languages 148 *Element presence* 148 *Context-sensitive checks* 148 *Reporting unknowns* 149 *It's only a beginning* 149 Example 3.3 149

chapter 4 *An overview of XSLT* 155

Attention: Version bump ahead 158

4.1 **XSLT history** 158

Limitations? What limitations? 159 *XPath as a guest musician* 160 *Beyond 1.0* 160

4.2 **A Gentle Introduction into 2.0** 161

Sequences 161 *Data types* 162 *Grouping* 162 *XPath 2.0* 163 *Extensibility* 164 *Multiple output documents* 164

4.3 **Taming a functional language** 165

Everything is possible by asking the right questions 165 *Why is XSLT functional?* 166 *Relearn, rethink, rewrite* 166 *Local variables* 166 *Passing parameters* 167 *Recursion* 167 *XPath tools* 168 *Temporary XML documents* 168 *Chaining templates* 169 *Extensions* 170

Example 4.1 167
Example 4.2 169

4.4 **XSLT extensions** 170

4.4.1 EXSLT 171

4.4.2 Saxon extensions 171

4.4.2.1 The assignability dilemma 172
4.4.2.2 Optimization-related extensions 173
4.4.2.3 Debugging extensions 174
4.4.2.4 Miscellaneous goodies 175

Nodesets 175 *PIs, DTDs, entities* 175 *Output control* 175 *Parsing strings and serializing trees* 175

4.4.3 Custom extensions 176

4.5 **Overview of an XSLT stylesheet** 176

4.5.1 Templates 177

Applicable vs. callable templates 177 *Push vs. pull templates* 178 *Trunk vs. branch templates* 178

4.5.2 Layout code 179

4.5.3 Variables and functions 180

Don't force users to edit the stylesheet 180 *Lazy evaluation* 180 *Functions vs. templates* 180

4.5.4 Control flow 181

4.5.5 Debugging and documentation 181

chapter 5 *The XSLT stylesheet* 183

Focusing on what's important 186 *Not only XSLT* 186

5.1 **Schematron validation** 186

5.1.1 Shared XSLT library 187

5.1.1.1 Stylesheet parameters 187

5.1.1.2 Master document access 188

Who am I speaking to? 188

5.1.1.3 Pathnames 189

Where am I? 190 *Language* 190 *Abbreviated location* 191 *No way* 192 *Follow the right paths* 192

5.1.1.4 Deabbreviation functions 192

External links 193 *Internal links* 193

5.1.2 Schematron wrapper for Saxon 195

Custom wrapper 195 *Deployment* 196 *Namespace aliasing* 197 Example 5.1 195

5.1.3 Advanced Schematron 197

5.1.3.1 Document availability 198

5.1.3.2 Internal links 200

5.1.3.3 External links 201

A quest for a better probe 201

5.1.3.4 Local images 202

5.1.3.5 Language links 203

5.1.3.6 External blocks 203

Orthogonal block definitions 203 *Dynamic block definitions* 204 *Block references* 204

5.1.3.7 Uniqueness 204

IDs in DTDs 204 *The Schematron way* 205 *Diagnostics* 206 *Extensible uniqueness* 206 *Stay tuned* 206

5.2 **Stylesheet: first steps** 207

5.2.1 Setting up the environment 207

Preliminaries 207 *Output settings* 207 Example 5.2 207

5.2.2 Page skeleton 208

Diagnostic output 210 *Page versions* 210 *Layout tips* 211 Example 5.3 208

5.2.3 Static templates 211

CSS 211 *The benefits of automation* 212 *JavaScript* 212 Example 5.4 212

5.2.4 Miscellaneous pieces, from title to footer 212

Title 212 *Metadata* 213 *Bottom stuff* 214

5.3 **Top-level structures** 214

5.3.1 Menu 215

Items as images 216 *Item states* 217 *The complexity range* 217 *Submenu organization* 218 *Static submenus* 218 *Orthogonal menus* 218 *Interactive submenus* 218 *Collapsing trees* 218 *Recursive menu* 219 — Example 5.5 215

5.3.2 Dynamic menus 219

5.3.2.1 Reading a directory with Java 219

String or nodeset? 219 *Filtering files* 221 — Example 5.6 220

5.3.2.2 Setting up Java extensions 221

Installing a class 221 *Plugging it in* 222

5.3.2.3 Example: linking files 222

5.3.3 Blocks 223

5.3.3.1 Orthogonal blocks 224

A template with multiple inputs 224 *Deabbreviation of orthogonal sources* 224 — Example 5.7 225

5.4 **Low-level structures** 225

One-to-one mapping 225 *Be careful with shadows* 225

5.4.1 Processing links 226

Example 5.8 226

5.4.2 Text processing 227

5.4.2.1 Charset conversions 227

5.4.2.2 Search and replace 228

The brute force approach 229 *The XSLT 1.0 approach* 229 *The XSLT 2.0 approach* 229 *The combined approach* 229 *Alternating quotes* 231 *Punctuation as style, not content* 232 — Example 5.9 230

5.4.2.3 Text preparation guidelines 232

Standardize 232 *Reuse* 233

5.4.2.4 Adding structure 233

Example 5.10 233

5.5 **Dealing with non-XML objects** 234

5.5.1 Accessing images 234

Checking existence 235 *Retrieving objects' properties* 235 *Getting dimensions* 235 *Image insertion template* 236 *Building an image gallery* 237 — Example 5.11 235, Example 5.12 237, Example 5.13 237

5.5.2 Creating images 238

Two-stage conversion 238 *Why put text into images?* 238

5.5.2.1 Choosing format 239

5.5.2.2 Choosing rasterizer 239

Batik 239 *Imagemagick* 240

5.5.2.3 Preparing fonts 240

PfaEdit 241

5.5.2.4 Creating SVG **241**

Another language to learn? **241** *Defining the canvas* **241** *Linking the font* **241** *Creating the text* **242** — Example 5.14 **242**

5.5.2.5 Running conversion **242**

Wow! **243** *Designers beware* **243** — Figure 5.1 **243**

5.5.2.6 Postprocessing **244**

Figure 5.2 **244**

5.5.2.7 Once more, with XSLT **244**

Java as a launchpad **244** *Image generation template* **245** *Not exactly fly* **247** — Example 5.15 **245** Example 5.16 **246**

5.5.3 Creating other binary formats **247**

5.5.3.1 Flash **247**

SWFML **247** *Ming* **248**

5.5.3.2 PDF **248**

TEX **249** *XSL-FO* **249**

5.5.3.3 Other formats **249**

5.6 **Batch processing 250**

Statement of the problem **250** *The importance of being functional* **250**

5.6.1 Launcher templates **251**

Launching transformation **251** *Other commands, other environments* **252** — Example 5.17 **252**

5.6.2 The batch template **253**

Other site-wide jobs **253** *Separation vs. convenience* **254** — Example 5.18 **253**

5.6.3 Problems and solutions **254**

Orthogonal to everything else in the stylesheet **254** *Portable* **254** *May take quite some time* **254** *Validation errors do not stop batch processing* **255**

5.7 **Summary examples 255**

5.7.1 Shared XSLT library **255**

Example 5.19 **255**

5.7.2 Advanced Schematron schema **259**

Example 5.20 **259**

5.7.3 Stylesheet **262**

Components **262** *Processor* **262** *Graphic software* **263** *Windows paths* **263** *Deployment* **263** *Layout overview* **265** *Don't ask me why* **266** — Figure 5.3 **265** Example 5.21 **266**

chapter 6 *XML software* 277

A no monster zone 280 *Bang for the buck* 280 *Standards compliance* 280

6.1 **Authoring XML** 281

Developer's workbench 281 *Author's writing desk* 282 *Editor's assembly line* 282 *What lies ahead* 283

6.1.1 Source editing 283

You can do it too 283

6.1.1.1 Features 283

Generic XML features 284 *Schema-specific features* 284 *Syntax coloring* 284 *XPath tools* 285 *External processing* 285 *Project management* 285

6.1.1.2 Examples 286

Emacs 286 *<oXygen/>* 287 *Collaborative Markup Editor* 290 *Transforming Editor* 290

Figure 6.1 288
Figure 6.2 289

6.1.2 Graphical XML editing 291

6.1.2.1 Tree metaphor 291

Figure 6.3 292

6.1.2.2 Frames metaphor 293

Pollo 293 *Overcaffeinated!* 295

Figure 6.4 294

6.1.2.3 Icon tags 295

Morphon 295

Figure 6.5 296

6.1.3 Form-based editing 296

Tabular forms 298 *Smart forms* 298

6.1.3.1 Using XForms 299

X-Smiles 299 *The role model* 303 *Loading and saving* 303 *Constraining input* 303 *Please type* 303 *Growing the document* 304 *When you're done* 304 *Limitations* 304

Example 6.1 300
Figure 6.6 305

6.1.4 Wordprocessor editing 306

Syntax formatting 306 *Cascading Style Sheets* 307 *Memory for faces and memory for names* 307 *What you see is what you pay for* 308 *XML comments* 308 *Attributes* 308 *Nesting of elements* 309

6.1.5 Writing CSS for XML visualization 309

But we already have a web site? 309 *HTML text in the XML structure* 310

6.1.5.1 Nested boxes 310

6.1.5.2 Links and images 310

Example 6.2 312

6.1.5.3 Example and demonstration 311

Figure 6.7 314

6.2 **Converting into XML** 311

Not a simple algorithm 312 *Low-level (syntactic) processing* 313 *High-level (semantic) processing* 313 *Multiple inputs, multiple outputs* 315

6.2.1 Plain text 315

txt2html 315 *Chaperon* 315 *txt2xml* 316

6.2.2 HTML 316

HTML Tidy 316 *Common to many toolchains* 316 *Forcibly tidy* 317

6.2.3 Office formats 317

Only if they are willing to listen 317 *Who is to blame?* 317 *What is to be done?* 318 *The reference implementation* 318

6.2.3.1 Converting via plain text 319

Extracting paragraphs 319

6.2.3.2 Converting via HTML 319

Microsoft HTML 319 *wv* 320 *Rigidly quirky* 320

6.2.3.3 Converting via XML 320

Le style est l'homme même 320 *Upcast* 321 *The law of inertia* 321 *Styles are flat* 321 *Write the Style Bible* 322 *Minimize the disruption* 322 *Redefining the "source"* 323

6.2.3.4 MS Office 2003 324

6.2.3.5 Other Offices 324

6.2.4 Semantic processing and XML-to-XML conversion 325

The right point to fork 325 *Refactoring strategies* 326 *Formatting-related markup* 326 *Position* 326 *Content* 327

6.3 **XML utilities** 327

6.3.1 XML diff tools 327

Don't touch what you can't parse 327

6.3.1.1 Diffs for viewing 328

diffmk 328

6.3.1.2 Diffs for patching 329

diffxml 329 *patchxml* 329 *xmldiff* 329 *4update* 330

6.3.2 XPath tools 330

6.3.2.1 Command line 330

xpath 330 *Context provided separately* 331

6.3.2.2 GUI 331

XPath Explorer 331 *Both matching and calculating* 331 *Query expanded* 331 *Generated paths* 332 Figure 6.8 332

6.3.2.3 Shell 333

xsh 333 *So long as it's hierarchical* 333 *Tab, complete that* 333

6.3.3 Grammar generation 334

DTDGenerator 334 *NekoDTD* 334

6.4 **XSLT tools** 335

6.4.1 Processors 335

Saxon and the rest 335 *The speed race* 335

6.4.1.1 Java 336

Xalan 336 *XSLTC* 336 *XT* 336 *Jd-xslt* 337

6.4.1.2 Python 337

4xslt 337

6.4.1.3 Native binary 337

Sablotron 337 *Libxslt* 337

6.4.2 Generators 338

Stylevision 338

6.4.3 IDEs 339

6.4.3.1 Processor-neutral 339

Treebeard 339

6.4.3.2 Processor-specific 340

XSLT-process 340 Figure 6.9 341

6.4.4 Profilers 340

catchXSL! 340 Figure 6.10 343

6.5 **Build tools** 342

Housekeeping chores 342 *Process what was changed* 344 *A hierarchy of tasks* 344 *Tools for building projects* 344

6.5.1 make 345

6.5.1.1 Validation and transformation script 345

One more layer of logic 346 *A language for scripting what?* 346 *Variables and parameters* 346 *Two-stage validation* 347 *No matter whose problem it is* 347 *If and only if* 347 Example 6.3 346

6.5.1.2 Makefile 347

Variables 348 *Upload rule* 350 *Smart uploading* 350 *Build rule* 350 *Page processing rule* 350 *Schema compilation rule* 351 *Cleanup rule* 351 Example 6.4 349

6.5.1.3 Running make 351

Explicit targets 352

6.5.1.4 Makefile generation 352

6.5.2 Apache Ant 353

Build file is XML 353 *Java classes, not OS commands* 353 *Targets only depend on other targets* 353

chapter 7 *XML on the server* 355

7.1 **XSLT processor as servlet** 358

7.1.1 Saxon servlet 358

Can we simplify this? 358

7.1.1.1 Tomcat servlet engine 359

7.2 **Apache Cocoon** 359

Cocoon step by step 360 *A bit of history* 360

7.2.1 Cocoon applicability 361

Enter logic 361

7.2.1.1 Dynamic sites with XML 361

Building blocks 362 *Java, again* 362

7.2.1.2 Server-side XML 362

The XSLT bottleneck 362 *Size matters* 362

7.2.1.3 Cocoon's downsides 363

Server (re)configuration required 363 *Big and complex* 363 *Unstable, evolving* 363

7.2.2 Pipelines 364

DOM vs. SAX 364 *Generators* 364 *Transformers* 365 *Serializers* 365 *Readers* 366 *Aggregators* 366 *Actions* 366

7.2.3 Sitemap 366

7.2.3.1 Matching requests to pipelines 366

Matcher 366 *Pipelining from afar* 367 *Readers for static content* 368 *Other wildcard options* 368 *Towards the extensionless Web* 369 *Virtual URIs vs. abbreviated addresses* 369

7.2.4 Aggregation 370

XInclude transformer 370 *CInclude transformer* 371 *Sitemap aggregator* 371

7.2.5 Dynamic processing 372

Legacy code 373 *XSP* 373 *Logicsheets* 374 *Actions* 374

7.2.6 document()ing in Cocoon 375

Stay in line 375 *Logic or style?* 376 *Forking trouble* 376

7.2.7 Cocoon primer 377

Example 7.1 379

7.2.8 Foobar under Cocoon 380

XSLT 2.0 and XPath 2.0 380 *Extension Java classes* 380 *Directories and URLs* 380 *Master document* 381 *Shared XSLT library* 381 *Schematron validation* 382 *Image generation* 382

Bibliography 385

Index 387