

Tim Kennedy
Mary Slowinski

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language" xmlns:m="http://features.real.com/2001/SMIL20/Extensions">
  <head>
    <meta name="title" content="Villa Travel Glass Blowing Trip" />
    <meta name="author" content="Tim Kennedy, Online Delivery" />
    <meta name="copyright" content="(c) Images and Video: Mary Slowinski, Music: Joodle Music/ASCAP" />
    <meta name="abstract" content="A multimedia look at Villa Travel's trip through glass blowing factories." />
```

SMIL

adding multimedia to the web

```
  <transition id="rightSlide" type="slideWipe" subtype="fromRight" />
  <transition id="leftSlide" type="slideWipe" subtype="fromLeft" />
  <transition id="fadeBrown" type="fade" subtype="fadeToColor" fadeColor="#CC9966" />
  <transition id="fadeOrange" type="fade" subtype="fadeToColor" fadeColor="#FFCC66" />
</head>
```

```
<body>
  <par dur="indefinite">
    <animation src="animation/VillaFlash.swf" begin="00:00.0" region="flash" fill="remove" />
    <brush id="rp1" color="#FFFFCC" transln="leftSlide" begin="00:01.5" region="rp" fill="hold" />
    <brush id="vid1" color="#FFCC66" transln="leftSlide" begin="00:06.5" region="vid" fill="hold" />
    <brush id="img1" color="#CC9966" transln="leftSlide" begin="00:07.5" region="img" fill="hold" />
    
    <textstream src="text.rtf" begin="00:08.0" region="text" fill="hold" />
  </par>
</body>
```

```
<excl dur="indefinite">
```

```
  <par begin="00:09.0">
    
    <param name="bitrate" value="23000" />
  </img>
  
  <param name="bitrate" value="23000" />
</img>
```

Amazon v. Audio Pod
US Patent 8,738,740
Amazon EX-1090

<http://www.SMILBook.com>

SMILBook.com is the companion Web site for *SMIL: Adding Multimedia to the Web*. It features online examples from the book, links to important SMIL resources, and news about the SMIL community.

To get connected with SMIL, start with
SMILBook.com.

Tim Kennedy

Mary Slowinski

SMIL:

Adding Multimedia to the Web



Copyright ©2002 by Sams Publishing

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

International Standard Book Number: 0-67232-167-x

Library of Congress Catalog Card Number: 00-109550

Printed in the United States of America

First Printing: November 2001

04 03 02 01 4 3 2 1

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Sams Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The authors and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of programs accompanying it.

Executive Editor

Jeff Schultz

Development Editor

Kate Small

Managing Editor

Charlotte Clapp

Project Editor

Elizabeth Finney

Copy Editor

Barbara Hacha

Indexer

Angie Bess

Proofreader

Mary Ann Abramson

Technical Editor

David Warner

Team Coordinator

Amy Patton

Media Developer

Dan Scherf

Interior Designer

Gary Adair

Cover Designer

Dan Armstrong

Page Layout

Rebecca Harmon

Lizbeth Patterson

Overview

Introduction 1

Part I Understanding SMIL: The Basics 5

- 1 What Is SMIL? 7
- 2 SMIL Authoring 19

Part II Using SMIL: The Specification Modules 37

- 3 Getting Technical: The SMIL Specification 39
- 4 First Things First: The SMIL Structure Module 49
- 5 Data About Data: The Metainformation Module 57
- 6 Placing It on the Page: The SMIL Layout Modules 63
- 7 Grouping Your Content: The Timing and Synchronization Module 85
- 8 Bring on the Media: The SMIL Media Object Modules 115
- 9 From This to That: The Transition Effects Module 133
- 10 Bringing Graphics to Life: The Animation Modules 157
- 11 Making it Interactive: The Linking Modules 189
- 12 Optimizing Content Delivery: The Content Control Modules 207
- 13 Speed Up, Slow Down and Do It Over Again: The Time Manipulations Module 227
- 14 RealNetwork's RealPix and RealText 235

Part III Using SMIL: The Projects 271

- 15 SMIL in RealPlayer 273
- 16 SMIL in Internet Explorer 305

Part IV Appendices 343

- A SMIL 2.0 Modules, Elements and Attributes 345
- B Online Resources 357
- Glossary 363
- Index 375

Contents

Introduction 1

Part I Understanding SMIL: The Basics 5

1 What Is SMIL 7

A First Look at SMIL	7
Why SMIL?	8
The Players and Browsers	10
RealPlayer	10
QuickTime	13
Internet Explorer	14
Other Players and Browsers	16
The Future of SMIL	17

2 SMIL Authoring 19

Authoring with SMIL	19
Basic SMIL Syntax	19
SMIL Document Concepts	20
Understanding Bandwidth and Data Rates	21
Calculating Bandwidth and Data Rate	24
Compensating for Bandwidth	26
Browser Compliance and Compatibility	27
Serving Your Work	27
SMIL Authoring Tools	31
GRiNS	32
Fluition	32
RealSlideshow	33
Macromedia Flash	34
HomeSite	35
Summary	35

Part II Using SMIL: The Specification Modules 37

3 Getting Technical: The SMIL Specification 39

The W3C Builds a Web Multimedia Language	39
SMIL 1.0	39
Timing Media	40
Media Layout	41
Old-Fashioned Linking	42
Anyone for a <switch>?	43
Limitations of SMIL 1.0	43

SMIL 2.0	44
SMIL 2.0 Modules	44
SMIL Profiles	45
Deprecation	46
Living in Multiple SMIL Worlds	46
On to the Specifications	47
4 First Things First: The SMIL Structure Module 49	
Conformance Profiles and the Structure Module	49
The Structure Module Elements	50
Element: <smil>	51
Element: <head>	51
Element: <body>	53
The Structure Module Attributes	54
Attribute: id	54
Attribute: class	54
Attribute: xml:lang	55
Attribute: title	55
Attribute: xmlns	55
Summary	56
5 Data About Data: The Metainformation Module 57	
The Metainformation Module Elements	57
Element: <meta>	58
Element: <metadata>	58
The Metainformation Module Attributes	59
Attribute: name	59
Attribute: content	61
Attribute: id	61
Attribute: skip-content	61
Summary	62
6 Placing It on the Page: The SMIL Layout Modules 63	
Conformance Profiles and the Layout Module	64
The Layout Module Elements	64
Element: <layout>	65
Element: <region>	66
Element: <regPoint>	67
Element: <root-layout>	69
Element: <topLayout>	69
The Layout Module Attributes	70
Attribute: backgroundColor	70
Attribute: background-color	71

Attribute: bottom, left, right, top	71
Attribute: close	72
Attribute: fit	73
Attribute: id	74
Attribute: height, width	74
Attribute: left (see bottom)	75
Attribute: open	75
Attribute: regAlign	75
Attribute: regPoint	76
Attribute: region	78
Attribute: regionName	78
Attribute: right (see bottom)	78
Attribute: showBackground	78
Attribute: soundLevel	79
Attribute: top (see bottom)	79
Attribute: type	79
Attribute: width (see height)	80
Attribute: z-index	80
Subregion Positioning and Attributes	81
Attribute: bottom, left, right, top, height, width	82
Attribute: backgroundColor	82
Attribute: fit	82
Attribute: z-index	82
The Layout Modules Summarized	83
7 Grouping Your Content: The Timing and Synchronization Module	85
Conformance Profiles and the Timing and Synchronization Module	86
The Timing and Synchronization Module Elements	86
Element: <seq>	86
Element: <par>	88
Element: <excl>	90
Element: <priorityClass>	92
The Timing and Synchronization Module Attributes	93
Attribute: begin and end	93
Attribute: dur	98
Attribute: min	99
Attribute: max	99
Attribute: endsync	100
Attribute: repeat	100
Attribute: repeatCount	101
Attribute: repeatDur	101

Attribute: fill	102
Attribute: fillDefault	103
Attribute: higher	103
Attribute: lower	104
Attribute: pauseDisplay	105
Attribute: peers	106
Attribute: restart	107
Attribute: restartDefault	108
Attribute: syncBehavior	108
Attribute: syncBehaviorDefault	109
Attribute: syncTolerance	109
Attribute: syncToleranceDefault	110
Attribute: syncMaster	111
Attribute: timeContainer	111
Attribute: timeAction	112
The Timing and Synchronization Module Summarized	114
8 Bring on the Media: The SMIL Media Object Modules	115
Conformance Profiles and the Media Object Modules	116
The Media Object Module Elements	116
Element: <animation>, <audio>, , <ref>, <text>, <textstream>, <video>	117
Element: <brush>	118
Element: <param>	118
The Media Object Module Attributes	120
Attribute: abstract	120
Attribute: alt	120
Attribute: author	121
Attribute: clipBegin, clip-begin, clipEnd, clip-end	121
Attribute: color	123
Attribute: copyright	123
Attribute: erase	124
Attribute: longdesc	124
Attribute: mediaRepeat	125
Attribute: name	125
Attribute: readIndex	126
Attribute: src	126
Attribute: sensitivity	127
Attribute: title	128
Attribute: type	128

Attribute: value	129
Attribute: valuetype	130
Attribute: xml:lang	131
The Media Object Modules Summarized	131
9 From This to That: The Transition Effects Module 133	
Conformance Profiles and the Transition Effects Module	136
The Transition Effects Module Elements	136
Element: <transition>	137
Element: <transitionFilter>	138
The Transition Module Attributes	138
Attribute: begin, end	139
Attribute: borderColor	139
Attribute: borderWidth	139
Attribute: by	140
Attribute: calcMode	140
Attribute: direction	141
Attribute: dur	142
Attribute: endProgress, startProgress	142
Attribute: fadeColor	142
Attribute: from, to	143
Attribute: href	143
Attribute: horzRepeat, vertRepeat	143
Attribute: id	145
Attribute: mode	145
Attribute: repeatCount	145
Attribute: repeatDur	145
Attribute: targetElement	146
Attribute: transIn, transOut	146
Attribute: type, subtype	149
Attribute: values	149
The Transitions Effects Module Summarized	150
Transition Types and Subtypes	151
10 Bringing Graphics to Life: The Animation Modules 157	
Conformance Profiles and the Animation Modules	157
BasicAnimation and SplineAnimation Module Elements	158
Element: <animate>	159
Element: <set>	159
Element: <animateMotion>	160
Element: <animateColor>	161
The BasicAnimation and SplineAnimation Module Attributes ..	162
Attribute: attributeName	162
Attribute: attributeType	163

Attribute: targetElement	163
Attribute: href	164
Attribute: actuate	165
Attribute: show	165
Attribute: type	165
Attribute: from	166
Attribute: to	167
Attribute: by	168
Attribute: values	169
Attribute: calcMode	170
Attribute: accumulate	171
Attribute: additive	173
Attribute: origin	175
Attribute: path	175
Attribute: keyTimes	177
Attribute: keySplines	178
The Animation Modules Summarized	187

11 Making It Interactive: The Linking Modules 189

Conformance Profiles and the Linking Modules	190
The Linking Module Elements	190
Element: <a>	191
Element: <anchor>	192
Element: <area>	192
The Linking Module Attributes	194
Attribute: accesskey	194
Attribute: actuate	195
Attribute: alt	195
Attribute: coords	196
Attribute: destinationLevel	198
Attribute: destinationPlaystate	199
Attribute: external	199
Attribute: fragment	200
Attribute: href	201
Attribute: nohref	202
Attribute: shape	202
Attribute: show	203
Attribute: sourceLevel	203
Attribute: sourcePlaystate	204
Attribute: tabIndex	204
Attribute: target	205
The Linking Modules Summarized	206

12 Optimizing Content Delivery: The Content Control Modules 207

Conformance Profiles and the Content Control Modules	208
The BasicContentControl Module Element and Attributes	209
Element: <switch>	210
System Test Attribute In-Line Use	212
Attribute: systemAudioDesc	212
Attribute: systemBitrate (system-bitrate)	213
Attribute: systemCaptions (system-captions)	214
Attribute: systemComponent	214
Attribute: systemCPU	215
Attribute: systemLanguage (system-language)	215
Attribute: systemOperatingSystem	216
Attribute: system-overdub-or-caption	217
Attribute: systemOverdubOrSubtitle	217
Attribute: systemRequired (system-required)	218
Attribute: systemScreenDepth (system-screen-depth)	218
Attribute: systemScreenSize (system-screen-size)	219
The CustomTestAttributes Module Elements and Attributes	220
Element: <customAttributes>	220
Element: <customTest>	221
Attribute: customTest	221
Attribute: defaultState	222
Attribute: override	223
Attribute: uid	223
The PrefetchControl Module Elements and Attributes	224
Element: <prefetch>	224
Attribute: bandwidth	225
Attribute: mediaSize	225
Attribute: mediaTime	225
The SkipContentControl Module Attribute	226
Attribute: skip-content	226
The Content Control Modules Summarized	226

13 Speed Up, Slow Down, and Do it Over Again: The Time Manipulations Module 227

Conformance Profiles and the Time Manipulations Module	227
The Time Manipulations Module Attributes	228
Attribute: speed	228
Attribute: accelerate	230
Attribute: decelerate	231
Attribute: autoReverse	232
The Time Manipulations Module Summarized	233

14 SMIL That Isn't SMIL: RealNetworks' RealPix and RealText	235
Compatibility	236
The RealPix Element	237
Element: <imfl>	238
Element: <head/>	239
Element: <image/>	239
Element: <fill/>	240
Element: <fadein/>	240
Element: <fadeout/>	240
Element: <crossfade/>	241
Element: <animate/>	241
Element: <wipe/>	241
Element: <viewchange/>	242
The RealPix Attributes	244
Attribute: aspect	244
Attribute: author	244
Attribute: background-color	245
Attribute: bitrate	245
Attribute: color	247
Attribute: copyright	247
Attribute: direction	247
Attribute: dstx, dsty, dstw, and dsth	248
Attribute: duration	249
Attribute: handle	249
Attribute: height and width	250
Attribute: maxfps	250
Attribute: mime	251
Attribute: name	251
Attribute: preroll	252
Attribute: size	252
Attribute: srcx, srcy, srcw, and srch	252
Attribute: start	253
Attribute: target	254
Attribute: timeformat	255
Attribute: title	255
Attribute: type	255
Attribute: url	256
The RealText Elements	257
Element: <window>	257
Element: <clear/>	258
Element: <pos/>	259
Element: <required>	259
Element: <time/>	259

Element: <tu> and <t1>	260
Element: 	260
Element: 	260
Element: <center>	261
Element: <hr/>	261
Element: <i>	261
Element: 	261
Element: 	262
Element: <p>	262
Element: <pre>	262
Element: <s>	262
Element: <u>	262
Element: 	263
Element: 	263
Element: <href>	263
The RealText Attributes	263
Attribute: begin and end	263
Attribute: bgcolor	264
Attribute: charset	264
Attribute: color	264
Attribute: command:	265
Attribute: crawlrate and scrollrate	265
Attribute: duration	265
Attribute: extraspaces	266
Attribute: face	266
Attribute: height or width	266
Attribute: link	267
Attribute: loop	267
Attribute: mailto:	267
Attribute: protocol:path	267
Attribute: size	268
Attribute: target	268
Attribute: type	268
Attribute: underline_hyperlinks	269
Attribute: version	269
Attribute: wordwrap	269
Attribute: x and y	270
RealPix and RealText Summarized	270

Part III Using SMIL: The Projects 271**15 SMIL in RealPlayer 273**

SMIL Through the RealPlayers	273
The Catch	276
Tutorial Goals	276
Building the Plan	277
SMIL Namespace and Structure	278
Adding the Metainformation	279
Creating the Layout	279
Adding the Flash Animation	281
Painting the Screen with <brush>	281
Adding <transition>	282
Building the RealPix File	286
Loading the Images into RealPix	287
Fades and Crossfades in RealPix	287
The Final RealPix	288
Referencing the RealPix in the SMIL	289
Adding the Video	290
RealPix Without RealPix	291
Interactivity with <excl>	292
Bringing Regions to the Top	295
Labeling the "Buttons"	295
Adding the Logo	296
Creating the RealText	296
Dropping in the RealText	297
Putting It on a Web Server	300
Putting It on a RealServer	301
The Final Presentation	302

16 SMIL in Internet Explorer 305

The Catch	306
The Beauty	306
About This Tutorial	307
Planning the Presentation	308
Roughing Out the Page	309
Getting Started with a Template	310
Putting Up the Structure	311
Adding Timing	313
Time to Add Time Containers	316
Decorating with Transitions	318

Matching the Planned Layout	324
Putting a Graphic in Motion	329
Building the Background	331
Dealing with Bandwidth	333
When in Doubt...Stall!	335
Final Code and Final Thoughts	337

Part IV Appendices 343

A SMIL 2.0 Modules, Elements, and Attributes 345

Structure Module (See Chapter 4)	345
Metainformation Module (See Chapter 5)	346
Layout Modules (See Chapter 6)	346
Timing and Synchronization Module (See Chapter 7)	347
Media Object Modules (See Chapter 8)	349
Transition Effects Module (See Chapter 9)	350
Animation Modules (See Chapter 10)	351
Linking Modules (See Chapter 11)	352
Content Control Modules (See Chapter 12)	353
Time Manipulations Module (See Chapter 13)	355

B Online Resources 357

The Definitive Authority: The World Wide Web Consortium (W3C) SMIL Pages	357
Companion Site for This Book	357
SMILBook	357
Downloadable Media Players	358
RealPlayer 8.0	358
QuickTime 5.0	358
Internet Explorer 5.5	358
GRiNS-SMIL 2 Player (Oratrix)	358
SOJA Cherbourg 2	358
Authoring Tools	359
RealSlideShow	359
Fluition	359
GRiNS	359
HomeSite (Allaire)	359
SMIL 2.0 Authoring Resources	359
JustSMIL (Streaming Media World)	359
HTML+Time 2.0	360

RealSystem Production Guide: Assembling a Presentation with SMIL 2.0	360
The CWI SMIL Page	360
Universal SMIL	360
Webopedia	360
Web Review Looks at the Future of SMIL	360
SMIL 2.0: Codeless Animation in HTML	360
SMIL 1.0 Resources	361
JustSMIL (Streaming Media World)	361
A Primer on SMIL	361
SMIL: Multimedia Markup	361
An Overview of Using SMIL 1.0	361
RealSystem Production Guide: Assembling a Presentation with SMIL 1.0	361
Apple QuickTime's SMIL 1.0 Pages	361
RealNetwork's SMIL 1.0 Tutorial	362
RealNetworks SMIL 1.0 Evaluator	362
WebReview SMIL 1.0 Articles	362
Some Additional SMIL 1.0 Online Tutorials	362

Glossary	363
-----------------	------------

Index	375
--------------	------------

Dedication

From Tim:

To Joy and Katelynn for your patience and love.

From Mary:

To Jasmine and Bryan for making time memorable.

Acknowledgments

We'd like to thank the editors of this book, Kate Small and Jeff Schultz, for working diligently and patiently with us to make sure this text came to light.

We'd also like to thank our colleagues and students at Bellevue Community College for being patient with us as well and for acting as sounding boards for some of the ideas and the structure for the book.

Last, but certainly not least, we thank our families who have supported us through late nights, early mornings, and those hair-pulling moments.

Tim Kennedy

Mary Slowinski

About the Authors

Tim Kennedy is the originator of Just SMIL (<http://JustSMIL.com>), the first major informational Web site about the SMIL language. His company, Online Delivery, created the original Streaming Media World (<http://www.streamingmediaworld.com>) site for Internet.com. His speaking appearances include informational sessions at Internet World and RealNetworks conferences. An enthusiast for all types of media production, he is currently Chair of the Media Communication and Technology program at Bellevue Community College.

Mary Slowinski is a full-time faculty member at Bellevue Community College (<http://www.bcc.ctc.edu>) instructing in Web design, digital video editing, and streaming technologies. In the summer, she teaches intensive workshops for teachers and other educators to assist them with mastering and incorporating new technologies into their course curriculum. Her other interests include distance speed skating, world music, and viewing vistas.

Tell Us What You Think!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

As an Executive Editor for Sams Publishing, I welcome your comments. You can e-mail, or write me directly to let me know what you did or didn't like about this book—as well as what we can do to make our books stronger.

Please note that I cannot help you with technical problems related to the topic of this book, and that because of the high volume of mail I receive, I might not be able to reply to every message.

When you write, please be sure to include this book's title and author as well as your name and phone or fax number. I will carefully review your comments and share them with the author and editors who worked on the book.

E-mail: m3feedback@sampublishing.com

Mail: Jeff Schultz
Executive Editor
Sams Publishing
201 West 103rd Street
Indianapolis, IN 46290 USA

Introduction

Is This Book for Me?

Is this a book I should bother with? You've got to be asking yourself that, especially because time is scarce and your bookshelves are probably already groaning with tech books. So we'll make our case and be brief.

This book is an exploration into the use of the Synchronized Multimedia Integration Language, better known as SMIL. SMIL versions 1.0 and 2.0 allow Web authors to weave together multimedia by writing XML-based markup similar to HTML. You can create powerhouse Web multimedia presentations using a simple text editor. Seems almost too good to be true, but with the backing of the World Wide Web Consortium (W3C, the organization that develops standards for the Web), SMIL is here—and here to stay.

So what can you do with it, you ask? You can place linkable hot spots on a video clip, display a rolling text document with hyperlinks and background music, or create a slideshow of still and moving images with narration and accompanying URLs. As a matter of fact, with the release of SMIL 2.0 in 2001 with its increased flexibility and modularization, the sky's the limit on the ways you can combine and synchronize media using this language.

What this book offers is the tools to write SMIL code. The first part introduces you to SMIL with a brief history of SMIL's development. The second part is a discussion of each module in the new 2.0 spec. Then the book quickly gets to the meat of the issue: how to use SMIL to create projects targeted to the various players. The projects section walks you step-by-step through the creation of a media presentation for each major player/browser. After your tutorials are

completed, you can post them on the Web and test them using various platforms and browsers, which will further your real-world understanding of the implementation of this language. And you'll have access to our updates on the companion Web site, where you can get news, updates and so forth.

So should you buy this book? If you are involved in the creation of Web pages or sites and you include (or intend to include) multimedia of any form—sound, images, text, hyperlinks, video, animation—then yes, you should buy this book. If you are a beginner who is working with HTML and is ready to test the multimedia waters, then yes, you should buy this book. If you are a developer who wants to see what all this SMIL hoopla is all about, then yes, you should buy this book.

This is the next multimedia wave. Jump on.

Prerequisite Skills

This book is not a primer on writing Web pages; to get the most from it, you should know HTML and have a basic level of Web mastery. The tutorials will take you from there.

System Requirements

Writing SMIL code itself requires no more than a simple text editor. However, the multimedia portions of the presentations require players. The Web site that accompanies this book contains links to the three major SMIL multimedia players as well as several lesser-known players. We've also included browsers, where appropriate. Be sure you have 30MB–50MB of disk space for the installation of this software.

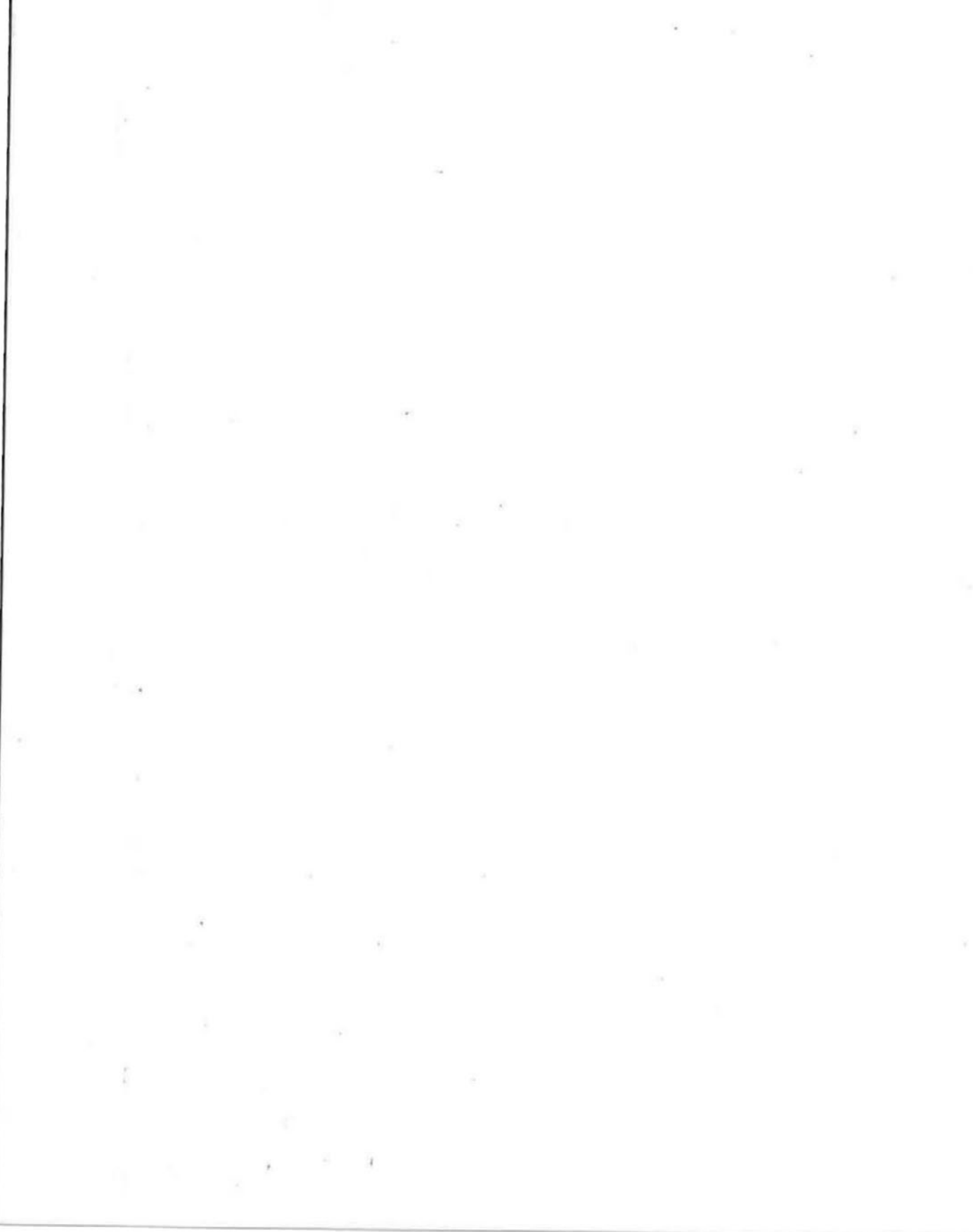
As for your machine, the faster the CPU and the bigger the hard drive the better. You'll need a minimum of 32MB of RAM. If you plan on working with your own multimedia source files, you'll need software to edit sound, video, and/or images. A graphics editing program would also be helpful.

Using the Book

Some people like to read background development stories, some do not. If you do, start with Chapter 1, “What Is SMIL?,” and go from there. If you don’t, you can go straight to the projects section. Then follow the instructions regarding the installation of the various players and browsers, and you will be good to go.

Additional Resources

This book, because of the nature of the content, is a work in progress. We’ve provided a Web site to accompany this book to keep our readers up to date on the developments that inevitably occur with relatively new technology. Feel free to visit us at <http://www.smilbook.com> to pick up additional authoring tips and to find out the latest SMIL developments.



Part I



Understanding SMIL: The Basics

1. What Is SMIL?
2. SMIL Authoring

Part 1

Understanding SML: The Basics

1. What is SML?
2. SML: A brief history



What Is SMIL?

In the world of Web authoring, HTML is the markup language that pulls everything together. Type a few markup statements into a text editor and you have a Web page that loads graphics and makes hyperlinks.

In the world of multimedia Web authoring, SMIL is the markup language that pulls everything together. Pronounced “smile,” it is an acronym for Synchronized Multimedia Integration Language. This friendly sounding, text-based markup language does just what it says: It synchronizes and integrates multimedia.

SMIL is a specification of the World Wide Web Consortium (W3C), a group of researchers and industry representatives that strives to advance the growth of the Web by developing standards. The instructions that make up SMIL are published in specifications maintained by the W3C. Based on XML, SMIL is a text-based language; therefore, you can use the same approach to authoring Web multimedia that you use to author traditional Web pages.

SMIL is still a young markup language. But it is also a powerful tool for creating captivating Web multimedia.

A First Look at SMIL

As a Web designer or developer, you are constantly inundated with complex new technologies. You probably grit your teeth and figure them out the best you can. Still, a bit of technology intimidation may exist that you might not be able to completely shake.

So as you begin this book on SMIL, we'll let you in on a little secret: SMIL is actually pretty easy. You can tell your colleagues that it is mind-blowing stuff—but we all know differently.

Like HTML, you can write SMIL with a basic text editor. As with all languages, SMIL continues to grow in complexity. But even so, you can master some basic SMIL markup with little effort.

The following is a short example of a basic SMIL document:

```
<smil>
<body>
  <par>
    <animation src="myflashfile.swf" />
    <audio src="myaudiofile.rm" />
  </par>
</body>
</smil>
```

In this example, I am playing a streaming Flash animation ("myflashfile.swf") in parallel (<par>) with a streaming audio file ("myaudiofile.rm"). At first glance, it might look pretty unfamiliar. But if you poke around in the markup, you might notice a few things.

You probably noticed that it is easy to type—just like HTML. You probably also noticed that the tags are similar or even the same as HTML tags you already know. And it is likely that you noticed that the tags reference external sources, the same as an HTML document might reference a graphic.

We will not try to mislead you. Like HTML, SMIL can get complex when you do complex things. But just as with HTML, you can do basic things with very basic markup language.

Why SMIL?

Why use SMIL? It is usually the first question asked. Compared to more established technologies and authoring environments such as Macromedia Flash, SMIL might seem to pale in comparison. What can SMIL do that Flash cannot?

We think that's really the wrong question. SMIL is not a competitive solution. It's not out to replace Flash or any other streaming-media architecture. Because SMIL is "technology agnostic," as shown in Figure 1.1, it allows Web multimedia technologies to collaborate with each other.

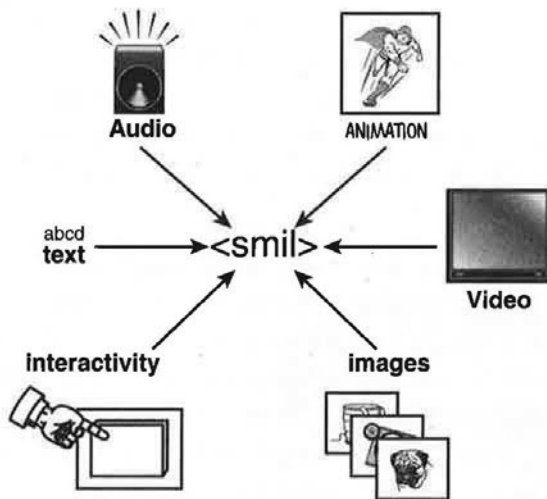


Figure 1.1 *SMIL can pull together a wide range of technologies.*

The growing Web multimedia market is a big business. A lot of dollars are up for grabs. That means that every involved company will continually push the Web multimedia envelope in pursuit of market domination. It would be nice to say that one technology will someday dominate. Then your authoring task would be neat and tidy. But the reality is that the competitive marketplace will always offer a range of technologies to choose from. A solution that helps you meld the strengths of these technologies into a cohesive presentation is just the kind of Web multimedia approach we all need. SMIL is that kind of solution.

Here is an example. Macromedia Flash is a great vector animation tool. It is acceptable for streaming audio but is not so robust with streaming video. RealNetworks' RealMedia is great at streaming audio and video. Why not combine the strengths of each? As illustrated in the code example in the previous section, SMIL allows that kind of collaboration to happen.

As a result, working with Web multimedia is like building a mosaic. SMIL acts as a glue that can hold it all together. But outside of its role as a streaming media referee, there are other good reasons for using SMIL.

The first is that because SMIL is text-based, it's easy for any designer or developer to work with it. Sure, you can buy a powerful SMIL authoring tool. As you'll see later in this book, a growing number are available to choose from. If you want to get started without any investment, however, the text editor that came with your computer will do.

Because SMIL is text based, the door is open for integration with other Web scripting and programming technologies. As long as that Web application can generate a text-based SMIL file, you can create a dynamic streaming presentation. For example, a shopper might fill out an HTML-based form and see a server-assembled Web multimedia presentation of products that match his or her criteria.

Finally, the SMIL effort is led by the W3C. As a group of researchers and industry representatives, the W3C is focused on ironing out standards that everyone can follow. Not all vendors will always agree with the direction that the SMIL specification may go. SMIL has already faced down challenges by major industry players. But as a standards body, the W3C tries to shape a specification that is beneficial to all parties involved. Unlike a proprietary technology owned by just one vendor, SMIL can lean toward general industry and consumer interests.

The Players and Browsers

One of the most amazing revolutions in Internet technology is happening rather quietly. In the early days of streaming, site visitors accessed Web multimedia with helper applications. These applications did little more than take a hand-off cue from the Web browser and play the content. They were accurately called *players*. Browsers browsed Web pages. Players played media content linked from those Web pages. The worlds were very separate.

Today, the differences between a player and a browser are getting pretty fuzzy. The newer player can load a URL directly. Your new player will probably handle traditional hypertext and images along with your streaming media. Your new player will likely let you take a look at the source markup file so that you can see how the Web multimedia presentation was assembled. Your new player might even display standard Web pages. Don't look now, but your new player is starting to sound a lot like a Web browser.

Let's take a look at the main contenders in the multimedia presentation race.

RealPlayer

As SMIL players go, RealNetworks' RealPlayer is one of the best-equipped Web multimedia players around. Officially, RealNetworks still refers to its application as "RealPlayer." "RealBrowser" would probably be a more accurate trade name. Like a Web browser, RealPlayer offers a window for entering URL addresses

directly (see Figure 1.2). Like a Web browser, RealPlayer allows users to view the markup source of presentations (see Figure 1.3). The RealPlayer format even allows authors the capability to encode meta information into their clips, making it possible for search engines to connect site visitors with presentations.



Figure 1.2 *RealPlayer in action. Komodo Dragon image ©2000 Online Delivery Inc.*

SMIL support started with the release of the G2 version of the player. At the time, it was the first major streaming player to support SMIL. RealPlayer supports a range of media technologies, including hypertext, still images, audio, animation, and video.

SMIL for RealPlayer is the most dominant use of SMIL today. SMIL has become a central piece of how RealNetworks pulls media technologies together. Through SMIL, RealNetworks content providers glue together animation, video, audio, and interactivity into finished Web multimedia applications.

RealPlayer supports multiple platforms and is available as a free download directly from RealNetworks. A RealPlayer Plus version with a few additional enhancements is available for a fee. For most SMIL situations, the free RealPlayer is sufficient for viewing and authoring.



Figure 1.3 You can view the source of the document in RealPlayer. Komodo Dragon image ©2000 Online Delivery Inc.

Currently, most tools that publish or author SMIL do so for RealPlayer. For example, Macromedia Flash exports SMIL to synchronize audio with Flash-based streaming animation. In this way, long animations with audio tracks can stay matched to the visual.

Currently, RealPlayer supports a set of proprietary SMIL-like markup languages for use with SMIL in their player. The two most common RealPlayer proprietary languages are RealPix and RealText. RealPix is a streaming image format that includes a variety of basic transitions and motion effects that can be rendered on-the-fly in RealPlayer. Combined with other media through SMIL, RealPix is a very effective means for delivering a quality multimedia experience on low-bandwidth connections. RealText is a hypertext format that offers scrolling, captioning, and other types of text support. Like SMIL, both RealPix and RealText can be written in a text editor using SMIL-like commands. They're then saved as separate documents that SMIL in turn references as the presentation plays. In the early days of SMIL support in RealPlayer, these RealNetworks proprietary languages were essential to round out RealPlayer's SMIL capabilities. As SMIL develops and there is a greater push toward SMIL that will run in any player, we may see the functions of these proprietary formats replaced by the modules in SMIL itself.

As with many newer Web applications, RealPlayer supports new technologies by automatically linking the site visitor with the appropriate plug-in to play the media content. This approach makes it easier for Web multimedia authors to use the latest features with less fear of losing the audience.

Because RealNetworks was the biggest early supporter of SMIL, it is important to keep an eye on how it is applying the language. Increasingly, RealNetworks seems to hint at its work in developing appliance- and database-driven uses of streaming media. How that will affect RealNetworks' use of SMIL is currently unknown, but it does open the door for exciting possibilities in dynamic Web multimedia authoring.

The latest SMIL 2.0 version of RealPlayer is called RealONE. The name change reflects a RealPlayer that is capable of handling a variety of multimedia playback, Web browsing, and desktop entertainment functions. As we write this, RealOne is currently available as a preview.

QuickTime

The implementation of SMIL in Apple Computer's QuickTime multimedia player, shown in Figure 1.4, started out modestly. Don't let that fool you. QuickTime has a long history in multimedia and supports a wide range of technologies. As a cross-platform player, Windows and Macintosh users alike can take advantage of QuickTime's features. Combined with SMIL, QuickTime makes for a very effective Web multimedia environment.



Figure 1.4 *The QuickTime player has VCR-like controls.*

Just how wide is the range of support for multimedia technologies in QuickTime? It would be an overstatement to say that QuickTime natively supports every digital media technology out there. But that overstatement would not be far from the truth. The QuickTime media architecture supports more than 200 digital media formats, supporting capabilities for audio, text, images, animation, music, interactivity, video, and virtual reality.

Of course, one of those supported formats is SMIL. After your SMIL is loaded, your presentation can call on media in any of those other 200+ formats that QuickTime supports. This alone makes QuickTime a SMIL powerhouse.

SMIL support in QuickTime first appeared in QuickTime 4.1. Initially, it was demonstrated as an easy way to add banner advertisements to a presentation. In this way, the banner ads could be easily changed without having to re-edit the main presentation. It is not a fancy approach to SMIL, but like QuickTime itself, it is solid and practical.

QuickTime system software, including a player, is available as a free download from Apple Computer for both Macintosh and Windows users. A QuickTime Pro version is handy for those authoring QuickTime-flavored SMIL presentations but is not essential for those simply playing presentations.

QuickTime still shows a strong emphasis on the player approach to Web multimedia. Like any Web browser, you can find a Load URL function under a pull-down menu in the QuickTime player. But with large controls and a TV-like appearance, the QuickTime player feels more like a desktop version of a video appliance than a Web browser. This is not a criticism. It is notable that this video-presentation philosophy is reflected in SMIL authoring tools geared toward QuickTime.

Internet Explorer

Although most other companies are approaching SMIL from the player side, Microsoft is approaching SMIL thoroughly from the browser side. Currently, you will not find Microsoft's SMIL support coming from the streaming media unit. Instead, as shown in Figure 1.5, SMIL has been gradually stepping up its appearance in Microsoft Internet Explorer since version 5 of the browser first appeared.

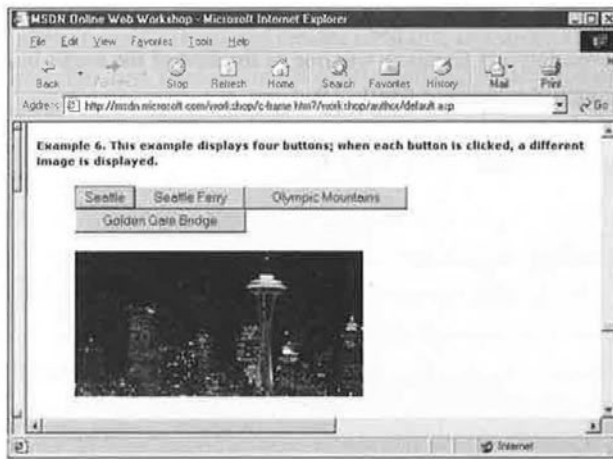


Figure 1.5 SMIL capability in Internet Explorer.

To understand Microsoft's approach to SMIL, a little history lesson is in order. Microsoft was initially a contributor to the first W3C-led industry effort to develop SMIL. At the last moment, Microsoft removed itself from the process as the W3C released SMIL 1.0. Depending on your world view, Microsoft made that step for either flattering or unflattering reasons. Nevertheless, it made its own SMIL proposal to the W3C a short time later. Initially called HTML+TIME (Timed Interactive Multimedia Extensions), Microsoft proposed a set of SMIL-based extensions to be used within an HTML Web document. Microsoft even released and later discontinued a companion product for the Office suite that could author Web pages in HTML+TIME.

Now in version 2.0, HTML+TIME is probably best described as special effects for the Web browser. With the release of SMIL version 2.0, Microsoft has rejoined the SMIL effort. It doesn't hurt that the SMIL 2.0 specification includes a subsection called HTML+SMIL that was written by the Microsoft representative to the SMIL effort.

It's hard to predict what the long-term prospects are for HTML+TIME or HTML+SMIL. As of press time, HTML+TIME is supported only in Windows versions of Internet Explorer. In that subset of the browser market, most of the support for HTML+TIME is found in Internet Explorer 5.5 or later. Without cross-platform support and limited to only the newest Internet Explorer Web

browsers, HTML+TIME is likely to be considered more of a gimmick for the short term. However, Internet Explorer was one of the first of the major browsers and players to support functions of SMIL 2.0.

Other Players and Browsers

After you get out of the big three (RealNetworks, Apple, and Microsoft), SMIL browsers and players tend to get more experimental and specialized. That should not discount the work that smaller companies are doing. Often, smaller organizations are the ones to push the envelope and show the bigger companies what can be done. However, you cannot expect the widespread adoption that a larger software publisher offers. These players and browsers offer specialized features that are often best limited to use for an intranet or a special purpose. With time, specialized SMIL players and browsers can come and go. The following are the most common that are continually offered.

GRiNS

The GRiNS player is one of the oldest SMIL players around. Short for Graphical Interface for SMIL, GRiNS was originally a research project of the National Research Institute of Mathematics and Computer Science in the Netherlands (known also as CWI). Now spun off as a product of a company called Oratrix, the GRiNS player fits into a larger commercial-product venture that includes a SMIL authoring tool. With its base in research, the GRiNS product typically supports newer versions of SMIL than its larger lumbering competitors do.

The GRiNS player is cross platform and is currently available as a free download from Oratrix (<http://www.oratrix.com/>). Support for various media types is usually accessed through the help of those players. So to view RealMedia in GRiNS, you need to have RealPlayer installed.

SOJA

What if your SMIL player was written in Java? Then you could serve the player and the content at the same time. That is the concept behind Helio SOJA, a Java-based SMIL player shown in Figure 1.6. Short for SMIL Output in Java Applets, SOJA is a convenient solution for content authors who want wider access to their content. Java applets are not always known for their robustness, and that probably limits the use of SOJA by a wider audience more than it should. SOJA is probably best used for SMIL presentations that are based on nonproprietary

Internet media. Helio, the creator of SOJA, is a small, French nonprofit organization. The Java files and tutorials are available for download at the Helio site (<http://www.helio.org/>).

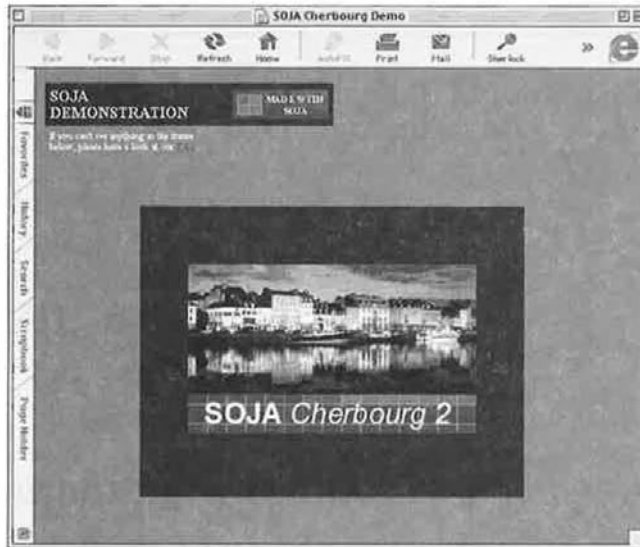


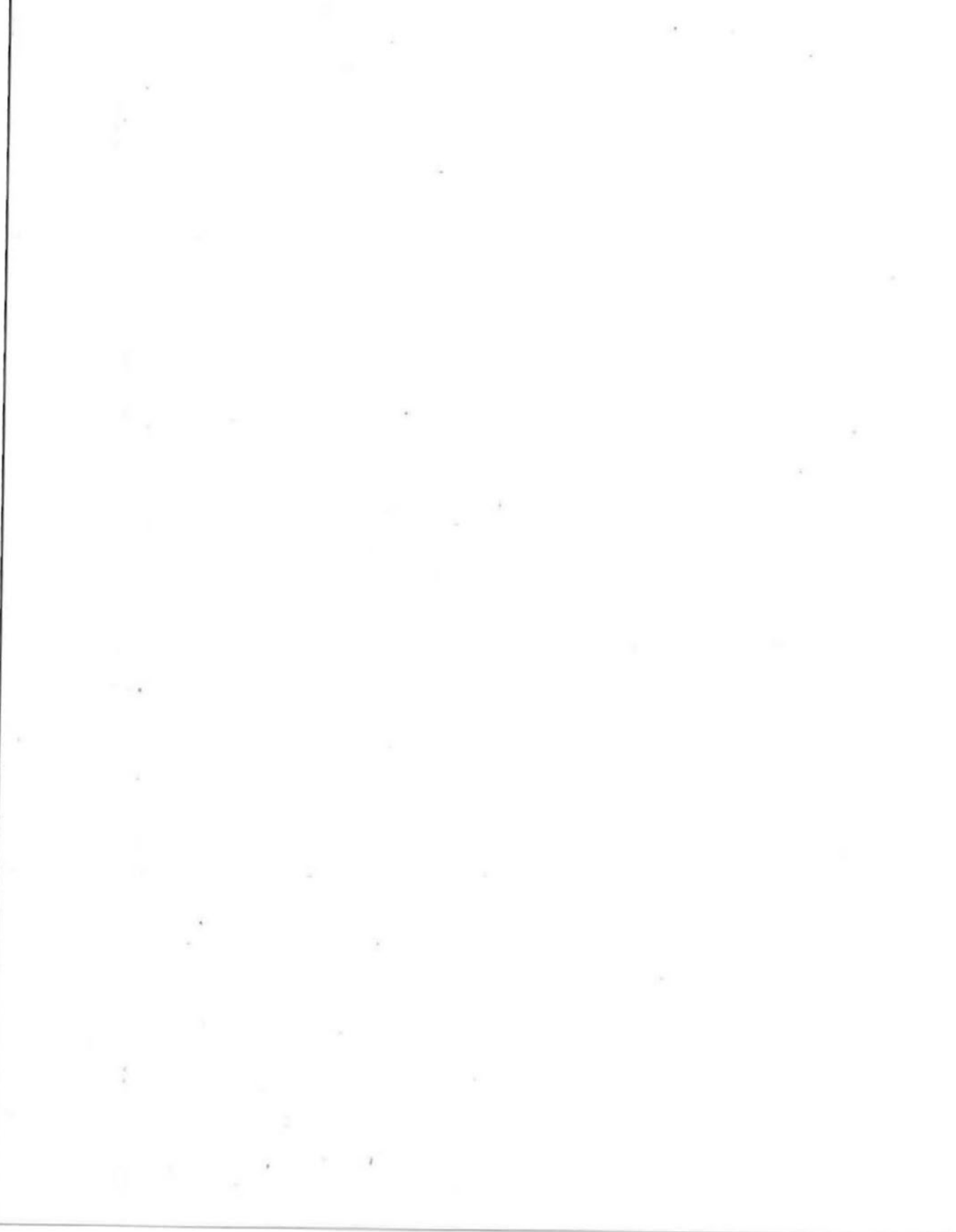
Figure 1.6 *SOJA is a Java SMIL player that plays right in the browser.*

Although Helio has been at it longer, it is not the only Java solution available as a SMIL player. SunTREC Salzburg offers Schmunzel, meaning “little smile” in German. It is no surprise that the home of Java, Sun Microsystems, is a partner in the technology and research center that offers this player.

The Future of SMIL

So what does the future hold for SMIL? If the last couple of years are any indication, SMIL has promise. Each year brings new tools and adopters. SMIL will not solve every problem faced by a Web multimedia author, but it can act as a facilitator to allow multiple media technologies to mix smoothly.

Now that you have a bit of SMIL background, we will dive in and take a closer look at the SMIL language and authoring multimedia for the Web.





SMIL Authoring

Authoring with SMIL

It is possible to hand-code your SMIL documents using nothing more than a text editor. In this section, we'll look at the basics of coding SMIL by hand, and in the next, we'll examine authoring tools that do the coding for you.

Basic SMIL Syntax

Writing SMIL is similar to writing HTML; in fact, if you feel comfortable writing HTML, you'll be off and running with SMIL in no time. By using a simple text editor (or any software that can save your work as a plain-text file) and following some basic rules, you can quickly produce a SMIL document that controls where, how, and when your multimedia clips are displayed.

In terms of coding, HTML can be quite forgiving. However, SMIL has some hard-and-fast rules you'll need to observe. Here are the basic rules of the road:

- All code must be written in lowercase. All attributes and tags must be coded in lowercase, without exception.
- Close every tag in the SMIL document, without exception. Some tags with a lot of content can be closed as in HTML; for example, `<smil>...</smil>`. Tags without a corresponding end tag must be closed with a forward slash at the end of the tag; for example, `<region .... />`.

- Attribute values must be enclosed in double quotes; for example, `<tag attribute="value" />`. Again, no exceptions.
- When stating values for attributes there should be no spaces around the equal sign. Adding spaces will create errors.
- SMIL documents are saved as `.smi` or `.smil` files; use the four-letter extension to decrease playback errors. No spaces are allowed in filenames, although an underscore can be employed. For example, `big bang.smil` is not allowed but `big_bang.smil` is acceptable.
- While not specifically a SMIL rule, it is a good idea to use the standard naming convention of eight characters or fewer followed by a dot and the three-letter file extension (for example, `rollaway.jpg`) when naming files. Again, no spaces are allowed in filenames.
- Recommended for ease when editing your code, but also not specifically a SMIL rule, is the use of HTML-style comments (`<!-- comments -->`) that are supported by SMIL, and indenting your code.

So those are the basic rules to keep in mind as you begin composing SMIL files. Other tag-related rules and restrictions exist (for example, the `<head>` section is optional, but the `<body>` section is required), but we'll cover those as we move through the specific tags and attributes in Part II.

SMIL Document Concepts

The structure of a SMIL document essentially revolves around two components: the `<layout>` section, which controls the positioning and display of the media elements, and the `<body>` section, which contains the paths to the media elements and the timing of their display.

A basic SMIL document structure is illustrated next:

```
<smil>
  <head>
    <layout>
      <!-- positioning and display -->
    </layout>
  </head>

  <body>
    <!-- media elements and timing -->
  </body>
</smil>
```

The `<head>` section of the document, in addition to outlining the layout of the presentation and defining how the elements are displayed, can also contain `<meta>` tags much like an HTML document does. These `<meta>` tags can include the presentation's title, author, copyright information, and an abstract describing the content. The `<head>` section is optional, but recommended, and is necessary for controlling the positioning of media elements on the screen.

The `<body>` section calls the media elements into the SMIL document by listing the path to the element and by assigning *handles*, or names, to the elements. Media elements can include live-streamed broadcasts, audio or video clips, text files, images, animations, streaming text and other media objects the player supports.

The timing and sequencing for these media elements also occurs within the `<body>` section. Elements can appear at the same time by grouping them using the parallel tag `<par>` or sequentially by using the `<seq>` tag. These tags can also be combined and nested as you like to present the media exactly as you want it displayed.

In addition to naming the elements and timing their appearance, the `<body>` section also provides several other options. For example, the `<switch>` tag, which allows the author to provide alternative presentations tailored to a range of options including the viewer's language or connection speed, is part of the `<body>` section. The capability to use an `<a>` tag to make elements or regions into links can also be coded into the `<body>` section.

We'll go into the details of these tags in Section II. Until then, just keep in mind that the `<layout>` section sets the stage, and the `<body>` section calls in the media.

Understanding Bandwidth and Data Rates

Bandwidth and Data Rate: the evil twins. Eternally linked and eternally battling.

That is one way to imagine the relationship between bandwidth and data rates. Here's another: Think of your presentation as a bucket of water being poured at a given rate through a hose into a basin. The basin represents the viewer of the presentation. The bucket of water is the file size of the total presentation (determined by adding together the sizes of all the component files in the presentation). The hose represents the available bandwidth. The rate at which it is poured through the hose equates to the average data rate. Data rate is calculated by

dividing that total file size (the bucket of water) by the duration of your presentation (the amount of time allowed for water to reach the basin). So, in other words, the data rate is how much water has to move through the hose in a fixed amount of time or, in this analogy, the rate of pour. See Figure 2.1 for a picture of this analogy.

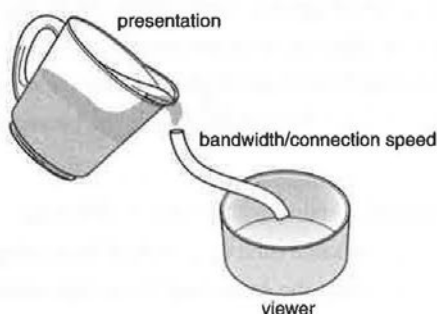


Figure 2.1 *A bandwidth and data rate analogy.*

So if you're trying to pour a big bucket of water very quickly through a drinking straw, all the water is not going to end up in the basin. As a matter of fact, much of the water is going to end up on the floor. That would be the case with a short-duration presentation with a big file size that is attempting to reach the viewer over a dial-up modem. Lots of water, coming in fast, through a very small hose.

This is the dilemma faced by Web multimedia authors because we tend to make large, beautiful presentations and then have to accept the bitter pill that most viewers are still viewing our work using the telephone lines.

Various solutions are available that, of course, all involve reducing data rate, not increasing bandwidth. We'll go into these solutions later. But no matter how you envision these two factors, this much is true: Bandwidth is almost always smaller than we would like, and data rates almost always require reduction. As a Web designer or developer, you're probably already familiar with these terms, but let's do a basic run-through just to make sure that we are all on the same page.

Bandwidth, the hose through which the media is poured, is the amount of data per second that the hardware, software, and connection are capable of delivering to the viewer. It is measured in kilobits per second (Kbps). For example, a

56Kbps modem is technically capable of delivering up to 56 kilobits of data per second.

Keep in mind that it is rare indeed for a viewer to connect at the top speed of a modem. Interference such as line noise, network congestion, and other sundry nuisances can reduce the connection speed, as can the need for error correction, resending lost data, and so on.

A good general rule is to expect only a portion of your viewer's bandwidth to be available for your presentation. For analog connections, such as dial-up modems, reduce the bandwidth figure by 25–30%. For digital connections (such as cable modems or DSL connections) reduce the bandwidth by 10–15%. In other words, assume that a 56Kbps modem offers 34–42Kbps connectivity, and a 256Kbps DSL modem reliably can provide bandwidth of 217–230Kbps (see Table 2.1). Then reduce your data rates accordingly. We'll talk more about that in a moment.

Table 2.1 Connection Speed and Available Bandwidth

Connection Speed/Type	Anticipated Available Bandwidth
28.8Kbps dial-up modem	20Kbps
56Kbps dial-up modem	34 – 42Kbps
128Kbps dual ISDN	90 – 100Kbps
256Kbps cable modem/DSL	217–230Kbps
512Kbps cable modem/DSL	435 – 460Kbps
T1 connection (at 1.5Mbps)	1275–1350Kbps

A file's *data rate*, measured in kilobytes per second, is the average amount of data per second that the file delivers. Determining a file's average data rate is simple: take the file size and divide by the file duration. For example, a 100KB file that takes 12 seconds to play has a data rate of 8.3KB per second. $\text{File size} \div \text{duration} = \text{average data rate}$. $100\text{KB} \div 12 \text{ seconds} = 8.3\text{K per second}$.

SMIL presentations typically include several files. To find the data rate for an entire presentation, take the file size of the individual elements and add them. Then divide the total file size by the overall duration of the presentation.

It's important to note that this is the *average* data rate of the file or presentation. In actuality, the data rate of a presentation fluctuates as elements begin or end, most notably when "heavyweight" elements (such as large files or sections of a file that include a lot of motion, transitions, or manipulations) enter or exit. Tools

are available that will assist you in visualizing and understanding your data rate so that you can create presentations that flow without clogging the pipe, so to speak. We'll talk about some of the methods for tinkering with this in "Compensating for Bandwidth," later in this chapter.

Now, back to the evil twins.

Calculating Bandwidth and Data Rate

Remember that bandwidth is measured in *kilobits* (Kb) per second and data rates in *kilobytes* (KB) per second. You'll have to convert one to the other to determine whether your data rate is exceeding your targeted bandwidth. We need to also take into account that although a modem Kb is a thousand bits, a file KB is 1,024 bytes. The math will not come out perfectly. So go for the ballpark figure, and leave wiggle room. That's the best you can do.

You can use several methods to convert between bits and bytes, bandwidth and data rates.

Let's use a standard example (see Figure 2.2) and run through a couple of ways to find out whether the presentation is going to stall.

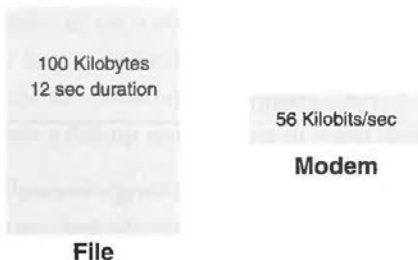


Figure 2.2 Our sample data used as an example for converting data rate and bandwidth figures for comparison.

Method 1

Let's say we've got a 100KB file that takes 12 seconds to play, and we are targeting a viewer with a 56Kbps modem.

1. First determine the presentation's average data rate. To do this, divide the file size by the presentation's duration. Using the

preceding example, our calculation would be $100\text{KB} \div 12 \text{ sec}$, which gives us an average data rate of 8.3 kilobytes per second.

2. Convert the data rate from bytes to bits. To compare our data rate to our connection speed, we need to multiply by 8 to convert from bytes to bits. The outcome ($8.3\text{Kbps} \times 8$) is 66.7 kilobits per second (or 66.7Kbps).

Because our example is using the connectivity of a 56Kbps modem, which allows true speeds of approximately 34–40Kbps (as noted in the previous section on adjusting for overhead), the file's data rate obviously surpasses the available bandwidth. There will either be buffering or lost data.

A quick-and-dirty alternative also is available for estimating whether your presentation's data rate will exceed the bandwidth. First you calculate the average data rate; in our example, it is 8.3Kbps. Then, using a factor of 10, convert the bandwidth kilobytes to kilobits. This makes the math easier to do and covers you for the potential loss of 25% connection speed because of network overhead. Thus, in our example, the viewer is on a 56Kbps modem. Dividing by 10 gives us a connectivity of 5.6 kilobytes for a 56Kbps modem. Because our average data rate for this example is 8.3Kbps and the bandwidth is limited to approximately 5.6Kbps, we come to the same conclusion—just more quickly.

Method 2

The following steps show another way to determine whether the bandwidth and the average data rate of your presentation are suited for one another:

1. First convert your file size from kilobytes (KB) to kilobits (Kb) by multiplying your presentation's total file size by 8. In our example, our 100KB file size would become 800Kb.
2. Divide the converted file size by the bandwidth to find the ideal duration. Again, using our example, for delivery on a 56K modem (and allowing for network overhead), you'd divide the file's size of 800Kb by 34Kbps to get an optimal presentation duration of 23.5 seconds. (File size in kilobits $\div$ connection speed = optimal duration. $800 \div 34 = 23.5$). The outcome is the ideal duration for the file so that it will play back smoothly and without any preloading or buffering.

Because our presentation is 12 seconds in duration, we have again reached the same conclusion. If we attempt to deliver our 100K, 12-second presentation over a 56K modem, there will be buffering, interruptions, and/or lost data because it needs 23.5 seconds to play in its entirety.

Compensating for Bandwidth

No matter which way you calculate it, our presentation example is not going to stream to our user without some delays or potentially lost data. Several things can help.

First, you can reduce the total file size of your presentation (reduce the amount of water in the bucket). Things such as transitions, high-quality images, video that includes a lot of motion, and stereo sound all can add to the file size of your presentation. Try reducing your file size by using the software you used to create the media to crop out unused space, reduce quality, reduce image size and/or frame rate, and eliminate filters and transitions. This is a good first option.

Another option to decrease your file size (which, in turn, reduces your data rate) is to increase the duration of the file (increase the time allowed for pouring). This is automatically done by some players, which won't begin playing a clip until enough data has been cached that will ensure uninterrupted playback (called *buffering* or *preroll*). The other way to extend the duration of your presentation is to add black or another screen matte to your timeline, thus inflating the duration.

You can also choose to begin a presentation with low-bandwidth clips, thus providing the audience with something to view while preloading is occurring. Appropriate choices for this may include simple text titles, a single 8-bit sound file, or a color field that slowly transitions to your first media clip. This is essentially a variant of creating a longer timeline. However, knowing that high-bandwidth clips within a presentation will cause your data rate to spike, you can use these techniques to smooth the way for preloading those high-density clips and smooth out those spikes in your data rate. See Figure 2.3 for a visual depiction of these concepts.

Last, but certainly not least, you can provide alternative versions of your presentation, specifically tailored for particular bandwidths. In some player formats, this has meant creating links to several separate presentations, but with SMIL, it is possible to assess the user's connection and launch the correct version for that connection speed. We'll talk over these options later on in Part III when we discuss the players in detail.

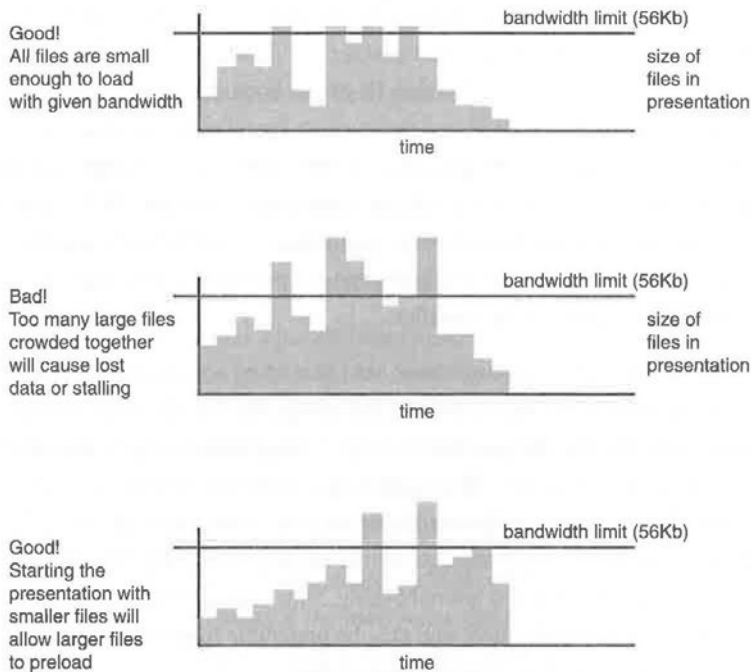


Figure 2.3 *A visual illustration of bandwidth limits and the loading of files.*

Browser Compliance and Compatibility

Although SMIL is a standard, because it is played back using proprietary players, the implementation of that standard becomes the discretion of the software publisher. For example, in the case of RealNetworks, RealPix and RealText are SMIL-based proprietary solutions. QuickTime, in turn, allows some attributes that RealPlayer would not be able to see. And Internet Explorer will do things that RealPlayer or any Web browser will not.

In most cases, Netscape and Internet Explorer will work with the players listed. Gradually, as implementations mature, there will be an increasing number of standard tricks to draw from. Until then, however, it is best if you target most content toward a particular player.

Serving Your Work

Let's say you've mastered SMIL (or perhaps one of the authoring tools) and have produced a completed presentation. How do you go about getting it out to your viewers?

Because RealNetworks, QuickTime, and Internet Explorer all handle the delivery of presentations in their own distinct manner, we'll cover the How-Tos for each when we dissect the players in Section III of this book. For now, suffice it to say that each is considered a *streaming architecture*, meaning that each delivery system contains a client-based player, an encoding scheme, and a server operating system that is distinct unto itself. All can make use of a standard Web server to deliver presentations, but do so in their own distinct manner. So for now, let's look at the general terms and conditions that are common to all, and then in Section III we'll return to the specifics.

First, let's define our terms. *Streaming* has come to be accepted as a rather broad term indicating that a viewer does not have to wait until a file has downloaded to begin viewing the file. Beyond that generality, there are actually two distinct types of streaming protocols. One is typically called "true streaming" with the other usually referred to as "pseudostreaming" or "progressive download." Neither true streaming nor pseudostreaming are superior to the other in every way. Depending on what you are publishing, who your audience is, and how they are accessing your presentation, one may be preferable to the other after you understand the strengths and weaknesses of each.

Here's a quick analogy to help get a picture of the differences between true and pseudostreaming or progressive download. True streaming is like watching a movie on television. It is broadcast, you watch it, and at the end of the show, it's gone. Pseudostreaming is more like watching a movie that is on television but catching it on your VCR. After it has been broadcast and taped, you can watch it again and again if you choose.

Pseudostreaming

Let's start our examination of these two types of streaming with streaming from a Web server. Streaming from a Web server is not considered true streaming because the streaming is not done in real-time. The file will take as long as it needs to completely arrive at the destination, regardless of the duration of the presentation. The two main benefits of this type of streaming are that the presentation received by the viewer can be of very high quality and almost all viewers can access the media. The downside is that the files download (to some extent at least), which takes up space on the viewer's machine, leaves your media at their disposal, and delivery times can be delayed by the protocols used to serve the

media. Depending on which architecture you are using, you can limit the extent of this downloading.

First, let's look at the how the file is transmitted. The data is served up using technology that is common for most Web delivery—a standard HTTP server using TCP/IP protocols (Transmission Control Protocol/Internet Protocol). This means that if viewers can access Web pages, they can probably access your presentation. Firewalls rarely present a problem with Web-based serving, which can be a definite plus depending on your target audience.

The TCP/IP protocol is a packet-based system of delivery. The data being sent using this protocol is broken up into small “packets” that are sent independently of one another and then knit back together when they are received by the client machine. The beauty of this system is a little item called CHECKSUM that makes sure all the sent data has arrived. If all packets and data are not accounted for, CHECKSUM signals to the sender to resend the entire file.

This means that all data will definitely arrive at the viewer's machine and makes it preferable when quality is the most important aspect of your presentation. The downside is that this can result in longer buffering or preroll for your presentations until all data is accounted for. Playback begins when enough of the file has been received to ensure constant playback without interruption, and if data is being resent, this can delay the start of playback while data is being cached.

Another downside with this type of serving is that it places the burden on the viewer. The file will, to some extent, download onto the viewer's machine, using hard drive space and leaving the viewer with a portion (or all) of your data. Also, because the amount of data being delivered is fixed, the connection speed of the viewer becomes the determining factor on how long the wait is to see the file (although you can provide alternatives from which the viewer can choose).

Good uses of pseudostreaming or progressive download, which take advantage of these strengths and weaknesses, include advertisements or movie trailers, where quality is key and missed data would be disadvantageous. In these cases, leaving the file on the user's machine for re-review would also be to the media creator's advantage.

True Streaming

In contrast, *true streaming* refers to media being streamed in real-time from a streaming server. The strengths of this type of streaming are the following: The

file is not downloaded onto the viewer's machine; The presentation can be automatically matched to the connection speed of the user; The media is received in real-time (meaning that a two-minute presentation will be presented in two minutes); and any point in the presentation can be randomly accessed. The downsides of true streaming are that it requires a streaming server and that data will be discarded if the viewer's bandwidth is insufficient to present all the data of your two-minute presentation in two minutes. Firewalls can also restrict access to files being served from a streaming server.

Files that are served from a streaming server frequently use UDP/IP (User Datagram Protocol), a form of transfer protocol that differs from TCP/IP in several ways. The most critical difference is that UDP does not use CHECKSUM to make sure all the data has been received. In fact, UDP allows for data to be dropped so that files will retain their original duration. In practice, this means that the quality of your presentation may be compromised, but the file will play exactly as long as it is supposed to. Preloading (often called buffering) may still occur, but the file itself, once started, will retain its authored duration. This is very helpful when trying to fit into a block of time or during live broadcasts.

One of the best features of true streaming is the capability to match the viewer's bandwidth and deliver the best quality possible, given this limitation. This is a function of the server's operating system, which first checks the connection speed and then delivers the appropriate file. Although the presentation quality may still be compromised, the viewer will get the best quality possible.

Good uses of true streaming, which take advantage of these strengths and weaknesses, include any live broadcast or situation where adherence to a broadcasting schedule is a reality. The fact that the files will retain their duration allows for a finite start and stop time and often, the files can be crafted so that not much data is dropped out.

Unfortunately, true streaming usually requires a streaming server. The cost of maintaining an additional server may not be an option for some multimedia authors. If not, it's back to Web-based, or pseudostreaming, serving!

Table 2.2 compares true streaming with HTTP streaming:

Table 2.2 True Streaming Versus HTTP Streaming

True Streaming	HTTP Streaming
Streaming server required.	Standard Web server.
UDP delivery protocol-data loss is acceptable and expected.	TCP delivery protocol - no data is lost but may need to be retransmitted, leading to slower delivery times.
Data is sometimes discarded to meet time and/or bandwidth constraints.	High-quality files can be guaranteed because no data is lost or discarded.
Can match user's bandwidth.	File is delivered without regard to user's bandwidth.
Burden on the developer (requires server, multiple versions).	Burden to the user (hard drive space, connection speed).
No file downloads to user's machine.	Files download to user's machine.
May not play behind some firewalls.	Bypasses most firewalls.
Delivers media in real - time - file will play exactly as long as it is supposed to, even if data gets discarded.	Playback will begin when enough of file has downloaded for uninterrupted playback and will for duration of file.

SMIL Authoring Tools

One of the biggest challenges to the adoption of SMIL has been finding good authoring tools. Sure, you can write SMIL in a text editor. But if you're going to author a lot of SMIL, you probably want some kind of authoring environment to make the process easier. Until recently, SMIL authoring tools were few and far between.

Thankfully, SMIL authoring options are starting to rapidly expand. A range of developing applications is now appearing on both the Macintosh and Windows platforms. It is still a little early yet, but the growth in SMIL tools is promising.

Next, let's take a look at some of the authoring tools currently available.

GRiNS

A player and an authoring tool? As you read in Chapter 1, “What is SMIL?”, GRiNS is a somewhat experimental but very capable SMIL player. On the authoring side, GRiNS is a capable and powerful SMIL authoring package (see Figure 2.4).

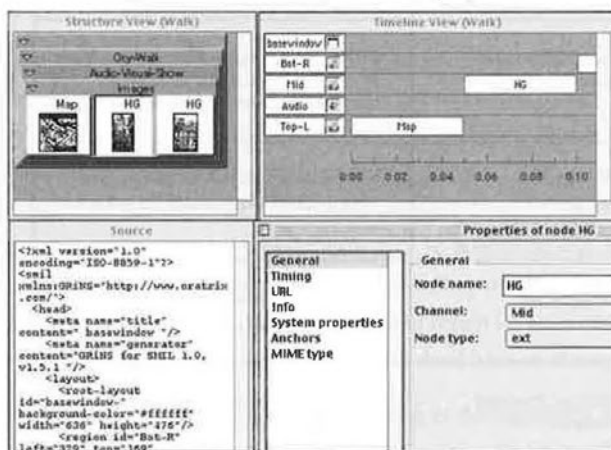


Figure 2.4 *GRiNS in action.*

GRiNS appeals to the multimedia engineer, meaning that it is more menu driven and a bit less visual than other tools. Using an object-oriented approach, the SMIL author can fine-tune a number of details within the presentation. With current versions, some familiarity with SMIL is important to get the most out of the authoring environment.

GRiNS is available from Oratrix (<http://www.oratrix.com>) for both the Macintosh and Windows environments.

Fluition

If GRiNS appeals to the multimedia engineer, Fluition is particularly adapted to the multimedia artist. Like GRiNS, Fluition is object oriented. But where the GRiNS environment leans toward menus and dialog boxes, Fluition is driven more by visual interaction with the author. Fluition, shown in Figure 2.5, is very well suited for SMIL presentations oriented toward video.

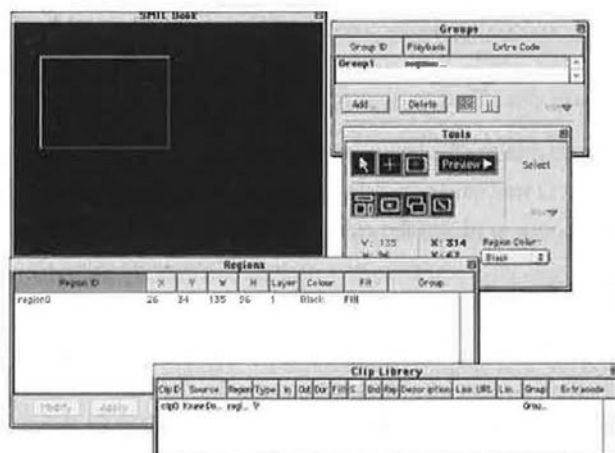


Figure 2.5 *Fluition in action.*

Fluition is available from Confluent Technologies

(<http://www.smilsoftware.com>) for both the Macintosh and Windows environments.

RealSlideshow

Of all SMIL authoring tools, RealSlideshow is one of our favorites. With other authoring tools, you can expect to have to spend some time getting used to how the application does business. This is not the case with RealSlideshow. As shown in Figure 2.6, the interface to RealSlideshow is simple and intuitive. After a couple minutes with the documentation, you can be up and authoring streaming-image presentations timed to music.

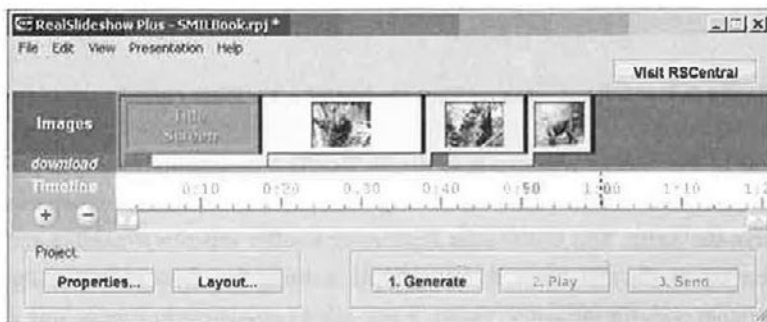


Figure 2.6 *RealSlideshow creates streaming slideshow presentations quickly and easily.*

RealSlideshow leans toward the SMIL-based proprietary RealPix and RealText approaches to authoring SMIL. With market dominance in the streaming-media market, RealNetworks can afford to do that.

RealSlideshow is available for the Windows platform as a free download from RealNetworks (<http://www.realnetworks.com/products/slideshow/>). Users willing to pay can choose to obtain the enhanced RealSlideshow Plus version.

Macromedia Flash

As an animation and Web multimedia tool, Macromedia Flash has wide acceptance and use and offers a range of output options for finished animation. So it was of great interest to the SMIL community when Macromedia, as shown in Figure 2.7, began offering SMIL support with Flash 5.

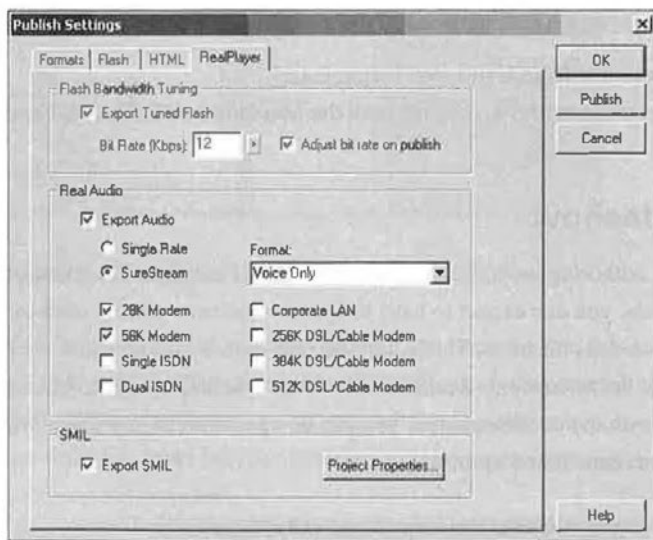


Figure 2.7 SMIL export capabilities are located in the Flash Publish options.

One of the available options is to stream the Flash animation from a RealNetworks RealServer. When using this option, the animation is separated from the audio. This enables the RealServer to offer superior streaming audio to accompany the animation. By using SMIL to bring the visual animation back together with the streaming audio, it also allows animators to stream long animations that maintain audio synchronization.

It is a bit of a stretch to call Flash a SMIL authoring tool. Flash currently offers little control over the SMIL output. But the prevalence of Flash makes the SMIL capability worth mentioning.

HomeSite

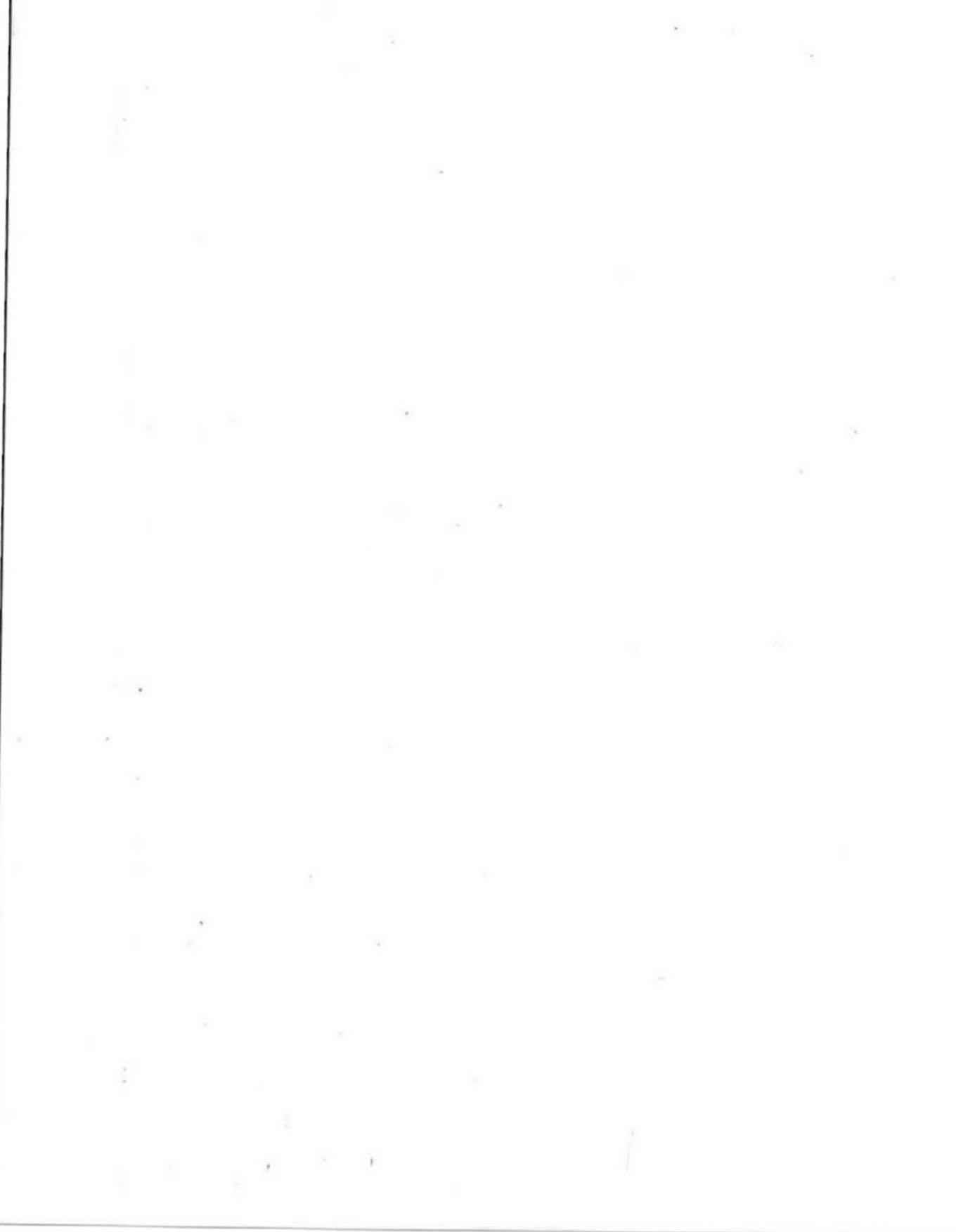
As the division between players and browsers blurs, so will the division between authoring HTML and authoring streaming-media presentations.

Allaire HomeSite is such a tool that blurs the difference. Long respected as an HTML authoring tool for Windows, HomeSite also supports output to SMIL.

When all is said and done, authoring tools for SMIL are on the rise. But for now, let's go forward and dig into coding this stuff ourselves using what we've learned so far.

Summary

SMIL is a versatile, powerful yet simple language you can use to author multimedia presentations for Web delivery. Authoring tools are rapidly being developed that will assist you with writing your SMIL presentation. This chapter gave you the simple guidelines for SMIL and streaming basics you will need to know before reading the more detailed discussion of how to author multimedia using SMIL 2.0, found in Part II of this book.





Part II

Using SMIL: The Specification Modules

3. Getting Technical: The SMIL Specification 39
4. First Things First: The SMIL Structure Module 49
5. Data About Data: The Metainformation Module 57
6. Placing it on the Page: The Layout Modules 63
7. Grouping Your Content: The Timing and Synchronization Module 85
8. Bring on the Media: The Media Object Modules 115
9. From This to That: The Transition Effects Module 133
10. Bringing Graphics to Life: The Animation Modules 157
11. Making it Interactive: The Linking Modules 189
12. Optimizing Content Delivery: The Content Control Modules 207
13. Speed Up, Slow Down, and Do it Over Again: The Time Manipulations Module 227
14. SMIL That Isn't SMIL: RealNetwork's RealPix and RealText 235



Getting Technical: The SMIL Specification

The W3C Builds a Web Multimedia Language

In October 1996, a small group representing the CD-ROM and Web multimedia communities gathered at a workshop hosted by the World Wide Web Consortium (W3C). Their task was to explore the creation of a language that could be used to synchronize multimedia on the Web. Unlike the scripting languages in use for creating multimedia, this new language would be descriptive. It would not require a programmer to make Web multimedia. Encouraged by the discussion, the W3C organized a working group to develop a specification for just such a language. From this working group came the Synchronized Multimedia Integration Language (SMIL).

SMIL 1.0

SMIL 1.0, the first specification of the Synchronized Multimedia Integration Language, was offered as a recommendation by the W3C in 1998 and is a model of simplicity. With only a few SMIL tags under your belt, you can easily create most SMIL presentations.

As stated in the specification, the goals of SMIL 1.0 were basic:

- To describe the temporal (time) behavior of the presentation
- To describe the layout of the presentation on the screen
- To associate hyperlinks with media objects

The resulting language is probably best summed up as HTML for multimedia. With SMIL 1.0, you have a text-based language that is built for multimedia on the Web. What really makes it powerful is that right out of the box, it comes ready to handle time. After years of using CD-ROM multimedia presentation approaches that had been roughly patched up for Web use, SMIL 1.0 showed a new direction.

Timing Media

SMIL 1.0 controls the timing of media with two basic elements: `<seq>` for the sequential synchronization of media objects and `<par>` for the parallel synchronization of media objects.

`<seq>` (for sequential) is used to play one piece of media right after another. With it, you can tell your SMIL player that you want your first media clip to be followed by a second.

`<par>` (for parallel) ensures that two independent pieces of media play at the same time. For example, using the `<par>` element, you can instruct a music clip to play in synchronization with a set of images.

Of course, you can combine these two basic building blocks in almost any fashion. For example, by nesting two `<par>` elements inside a `<seq>` element, one grouping of media clips running at the same time will be followed by a second block of assorted media objects also running in parallel. See Figure 3.1 for an illustration.

The capability to give the viewer a cohesive presentation from assorted media objects is incredibly important when evaluating SMIL as a potential Web multimedia approach. With other Web multimedia environments, the presentation of media is authored and output as a fixed work. If you want to provide other playback options using those same media objects, you need to recombine and output your content to match your intent. An example of a fixed Web multimedia would

be a QuickTime movie that includes an interactive Flash layer. Once the final movie file is created, the interactivity is fixed. Rearranging the media objects and changing the interactivity requires re-authoring the file.

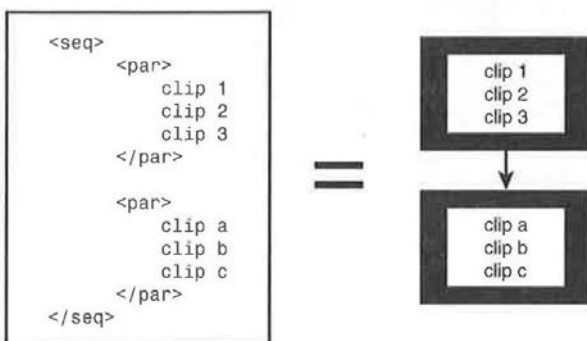


Figure 3.1 In this example, each block of media clips will play in parallel, but the first block (clips 1-3) will play before the second (clips a-c).

With SMIL, all the media objects maintain their status as independent media objects that can be pulled together as needed. For example, if you desire a different look for your presentation, a simple rearrangement of the `<par>` and `<seq>` elements in the SMIL document can easily achieve this effect. So either by your hand or automatically by a program running on a server, the presentation's organization and use of available media objects can be changed instantly to suit your intent, the viewer's hardware and software parameters, or the viewer's interests.

Media Layout

Timing and synchronization of media objects is very important, but that is only half the problem. HTML authors have long struggled with how to get Web pages to look the way they want them to. Thankfully, SMIL comes with built-in layout control.

The `<layout>` element determines how objects are displayed in the presentation space. Within the `<layout>` element, playback areas can be specified using the `<region>` tag. For example, suppose that you want to display video at the top of the presentation window accompanied by descriptive text at the bottom as in Figure 3.2. The SMIL `<layout>` element can be employed to create two regions: one for the playback of the video segment and another to display the text.

```

<layout>
  <region id="video" top="50px" left="100px" width="240px" height="180px"/>
  <region id="text" top="360px" left="50px" width="350px" height="20px"/>
</layout>

```

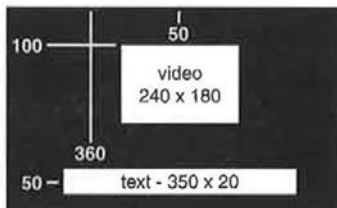


Figure 3.2 *Defining playback areas using <layout> and <region>.*

The <layout> element adds to the object-oriented power of the SMIL timing and synchronization elements. To reach our goal in the preceding example (video accompanied by text) using other multimedia authoring tools, it would be necessary to “burn” the descriptive caption text into the video and output it as such. If you wanted to change that text, you would have to go back to the original source files and create a new version of your video. With SMIL, the descriptive text can be kept as a separate media object and positioned in a <region> below the video with <layout>. Now, changing the captioning requires modifying only your text.

Old-Fashioned Linking

Of course, multimedia on the Web is pretty limited if it stays only within the confines of one start-to-finish presentation. As a delivery mechanism, CD-ROM is currently still more effective at delivering multimedia efficiently. A Web multimedia language needs to offer a functionality that cannot be contained on a single CD-ROM: hyperlinking to other materials and presentations.

SMIL 1.0 did not disappoint. Like HTML, SMIL offers the capability to link through an anchor element to another Web element. Notice the term “Web element” instead of “media element.” With a Web-based presentation, a SMIL object can link to another media object, but it can just as easily link to a Web page.

SMIL 1.0 also offers internal anchors. In this fashion, a link can jump within a SMIL document much like an internal reference in a long HTML page might move you down to a specific chapter.

Anyone for a <switch>?

Serving Web multimedia to an audience that might come to your presentation from a variety of places with a variety of equipment has the potential for creating authoring headaches. HTML authors have long faced this challenge and have used client-side scripting and server-side detection to offer up content that best matches the individual situations of their site visitors.

To help solve this problem for the multimedia author, SMIL offers the <switch> element. Do you have a visitor who speaks French? Give them a French version of your presentation with <switch>. Is the visitor using a slow modem or speeding along with a broadband connection? Determine the data rate and <switch> to the proper media files to match. Now site visitors can receive content tailored to their personal situations.

The <switch> element is indispensable in allowing the author freedom to work directly within the parameters of a specific audience member.

Limitations of SMIL 1.0

The SMIL 1.0 specification was a good starting point for a fledgling Web multimedia language. It is simple enough to learn quickly and generic enough that it can be adapted to almost any Web multimedia player. But, like any first effort, limitations existed.

RealNetworks enthusiastically jumped out front as the first major player to adopt SMIL. To get the most out of its G2 player, however, RealNetworks created two SMIL-based proprietary languages that met the need for low-bandwidth display of streaming text and images. RealText and RealPix provide text effects and transitions not available when using standard SMIL 1.0.

Whereas RealNetworks was a solid advocate for the first SMIL specification, Microsoft responded with criticism. Pulling out of the SMIL effort before the SMIL 1.0 recommendation was made by the W3C, Microsoft later offered a proposal to incorporate SMIL in with standard HTML.

These new interests would require a more powerful and complex language. SMIL 1.0 needed an update.

SMIL 2.0

Named after the city in which it was drafted, SMIL Boston first appeared during the summer of 1999. Later renamed SMIL 2.0, this specification was designed to meet two goals:

- To continue defining an XML-based language that controls the timing and positioning of media objects in a Web multimedia presentation
- To allow use of the SMIL syntax in other XML-based languages, such as eXtensible HTML (XHTML) and Scalable Vector Graphics (SVG)

The latter goal was extremely important to Microsoft because it addresses Microsoft's interest in incorporating SMIL timing elements into HTML and making them available to a Web browser. It also opens the door to much tighter incorporation of SMIL with other technologies, such as the emerging SVG format for graphics and animation.

SMIL 2.0 Modules

Whereas SMIL 1.0 is simple and easy to get into quickly, SMIL 2.0 is more complex—yet much more powerful. Divided into 10 functional areas that contain related modules, the SMIL 2.0 specification includes the following:

- Animation modules—These modules include elements for placing animation into the presentation timeline. They also allow for combining the effects of multiple animations.
- Content Control modules—These modules expand on the <switch> element and increase the control of presentation flow and format.
- Layout modules—These modules slightly augment the initial <layout> capabilities presented in the SMIL 1.0 specification. Additional support for audio layout is included.
- Linking modules—These modules amplify the hyperlinking capability of SMIL 1.0.
- Media Object modules—These modules further enhance the calling of media objects (such as animation, audio, video, text) as first defined in SMIL 1.0.
- Metainformation module—This module adds more descriptive power to the initial limited <meta> element that was offered in SMIL 1.0.
- Structure module—The basic elements of a SMIL 1.0 document gain additional attributes with the introduction of this module, which allows greater presentation control.

- **Timing and Synchronization module**—Building on the previous `<seq>` and `<par>` elements, the SMIL 2.0 Timing and Synchronization module offers more timeline control. The new `<excl>` element for playing an exclusive set of objects is also included here.
- **Time Manipulations module**—This module offers control of such time characteristics as the speed or rate of a media object. This is a particularly important module for implementing animation in the timeline.
- **Transition Effects module**—Absent from SMIL 1.0 was the capability to transition between one media object to another. This module adds the capability to offer standard wipes and fades between media objects.

SMIL Profiles

As you have seen, the SMIL 2.0 modules offer more elements and attributes than the simple capabilities available in SMIL 1.0. But the new specification does not stop there. SMIL 2.0 adds *profiles*, which further enhance the potential for smooth playback of multimedia presentations. Profiles combine modules as an attempt to achieve interoperability across players and browsers. Theoretically, presentations meeting established profile characteristics will be able to hop from player to player with acceptable results. But for the near term, most authors will probably still create presentations for specific players.

Currently, three language profiles make use of SMIL 2.0 modules. Two are host-language conformant, indicating that the profile incorporates a Structure module that contains the root element of the language. (In this case, that would be the `<smil>` element.) The main profile is called the SMIL 2.0 Language Profile. A subset of this profile is the SMIL 2.0 Basic Language Profile, which is basically a stripped-down profile for use with personal digital assistants (PDAs), mobile phones, and entertainment devices such as MP3 players. Yep. Portable MP3 players with image displays can SMIL.

The third language profile, the XHTML+SMIL 2.0 Language Profile, is based on a proposal by Microsoft and is undergoing development by the W3C. Although it is a subset of the larger SMIL 2.0 Language Profile, it is integration-set conformant, in contrast to host-language conformant. This means that the profile does not include the root (or `<smil>`) element. With this profile, SMIL is incorporated into the Web page along with HTML. In the past, Microsoft has referred to this approach as HTML+TIME. Extensions to HTML allow for timing and interactivity right within the HTML without using scripting languages.

Deprecation

As can be expected, anytime a language changes, some of its syntax also changes. Deprecation occurs when a portion of the language is no longer included or is marked for phasing out in a future specification.

Thankfully, deprecation between SMIL 1.0 and 2.0 is reasonably digestible. The most notable change between SMIL 1.0 and 2.0 is in the replacement of hyphenated attributes with mixed case (sometimes called *camel case*) attributes. For example, a `clip-begin` attribute would now be `clipBegin`.

Living in Multiple SMIL Worlds

All this talk of specification versions, modules, profiles, and deprecation may have you asking the ultimate SMIL question: Whom do I author for and with what approach?

Any kind of Web multimedia carries with it inherent risk and responsibility. For too long, Web multimedia designers and developers carried a self-righteous attitude about making content available for their visitors. If a viewer did not have the right browser or version of a player to interpret the media, they were doomed to put up with glitches, at best, or no content, at worst. In the lightning growth days of the Internet as a new medium, this attitude was tolerated as the norm.

Today, a designer or developer with that kind of attitude is best referred to as a fool. No company or organization hosting a Web site can afford to turn away potential customers or supporters, and any authors worth their salt will endeavor to meet the parameters set by their visitor's hardware and software. After all, every visitor can bring a sale to the bottom line. Over time, those little sales add up.

So how does Web multimedia fit in given all these potential constraints? Our suggestion is to use it with care and to consider the implications caused by the visitor who cannot see or hear what you create. Every authoring choice should help the site reach its objective or goal. If your Web presentation is mission critical, be sure to ask this question: Is Web multimedia the best way to deliver your content?

Likewise, think carefully about which version of SMIL you use. As we write this book, RealNetworks' RealPlayer and Apple's QuickTime offer solid SMIL 1.0

support, and playback using those players is relatively dependable. However, RealPlayer SMIL does not necessarily work with QuickTime SMIL. As for SMIL 2.0, formal adoption is on the horizon. Microsoft's Internet Explorer on the Windows platform supports some aspects of SMIL 2.0. RealPlayer and QuickTime support is right behind. But even as these new players become available, a large number of people on the Web will still be sticking with their older players for the time being, so SMIL 1.0 still presents a viable option. In any case, you will inevitably excite and motivate some viewers with the multimedia authoring choices you make. And you will leave others behind.

Although we certainly are not discouraging the use of Web multimedia, we do encourage exercising some common sense. For example, when considering viewer accessibility and authoring for a training-department intranet (with control over the access machines), the limitations of using SMIL 2.0 in Microsoft Internet Explorer may not be problematic at all. On the wide-open Internet, however, you would not want to limit your viewers to those using Internet Explorer on WinOS machines. RealPlayer with SMIL 1.0 support is currently common across multiple platforms and boasts a large number of users, so this may be the most practical authoring environment for your presentation. At the very least, you should provide multiple paths to your content so that unequipped users do not feel left out. True, they may not get the compelling Web multimedia touch that you have crafted for your luckiest visitors. But those left out are not as likely to leave your site frustrated or annoyed, either.

On to the Specifications

In the following chapters, we walk you through a closer look at the SMIL specification. Working from the outside to the inside of SMIL document construction, we explain the possible elements and their attributes. For your convenience, we highlight differences between SMIL 1.0 and 2.0 elements and attributes. We also inform you of which modules fit within which profiles. Finally, we cover player and browser support based on the current and beta software available at the time that we write this book.

SMIL is a living language that is constantly changing and adapting to the needs of the Web multimedia community. This book is a fixed record of that process. For the very latest news on SMIL, visit World Wide Web Consortium's SMIL information site (<http://www.w3.org/AudioVideo/>) and the companion site for

this book: SMILBook.com (<http://www.SMILBook.com/>). The W3C is the final authority on the SMIL specification, and SMILBook.com is a great way to keep up on how that is impacting the SMIL community.



First Things First: The SMIL Structure Module

The Structure module, with its `<smil>`, `<head>`, and `<body>` elements, provides the SMIL package to send your multimedia on its way. Put another way, the Structure module is the wrapper that surrounds all your additional coding. For the most part, you will not find a lot of complexity in this module. The elements it contains were introduced in SMIL 1.0, and because the Structure module of SMIL 2.0 makes use of these same elements, it is essentially backward compatible with SMIL 1.0.

Indeed, despite numerous changes between SMIL 1.0 and 2.0, the code for the structure of your document will stay pretty much the same as it has always been, with some small refinements.

Conformance Profiles and the Structure Module

As explained in the previous chapter, SMIL 2.0 introduces the concept of profiles. *Profiles* combine the various modules to increase interoperability across players and browsers. Presentations meeting profile characteristics should be able to hop from player to player with reasonable results.

Also explained previously are the two host-language conformant profiles (the SMIL 2.0 profile and SMIL 2.0 Basic profile) and one integration-set conformant

profile (the XHTML+SMIL 2.0 profile) that currently make use of the SMIL 2.0 modules. Host-language conformant profiles require a Structure module, which contains the `<smil>` element. The integration-set conformant profile does not include the `<smil>` element as a root element; instead, SMIL coding is incorporated into the Web page along with HTML.

As a result, the Structure module is included only in profiles that are SMIL host-language conformant. That means that although you will find the Structure module in SMIL 2.0 and SMIL 2.0 Basic profiles, you will not see it in SMIL Integration Set implementations such as XHTML+SMIL.

That may sound like mind-numbing technobabble, but it actually makes sense if you think about it. With an Integration Set implementation, SMIL is inserted into the markup of a Web page. You would not want the `<body>` and `<head>` elements of the SMIL Structure module to get in the way of the existing `<body>` and `<head>` elements of the Web page in which they sit. For this reason, the Structure module is not included in integration-set conformant profile presentations.

The Structure Module Elements

- Module: Structure
- Conformance: SMIL Host Language
- Profile Compatibility: SMIL 1.0, SMIL 2.0, SMIL 2.0 Basic
- Possible Elements: `<smil>`, `<head>`, `<body>`

As mentioned previously, the use of the Structure module `<smil>`, `<head>`, and `<body>` elements is quite similar to their use in SMIL 1.0. In fact, other than how Structure is written up in the documentation for the two versions, SMIL authors will not see a big difference between SMIL 1.0 and SMIL 2.0.

However, SMIL 2.0 does provide additional attributes for these elements, and these attributes, in turn, offer more choices and control for the author. We'll get to those in a minute.

First, let's take a minute to review what a basic SMIL document looks like:

```
<smil>
<head>
...This is where the markup for information about your content goes...
</head>
```

```
<body>
...This is where the markup for the presentation goes...
</body>
</smil>
```

For conventional Web authors, this simple SMIL Structure should look pretty similar to good old-fashioned HTML. Of course, there is more detail to the Structure module than this code snippet provides. If you understand the concept shown, you are almost ready to jump to the next chapter. But first take a closer look at the elements and attributes of the SMIL 2.0 Structure module.

Element: <smil>

- Version Compatibility: SMIL 1.0, SMIL 2.0
- Possible Attributes: id, class, xml:lang, title, xmlns
- Child Elements: <head>, <body>

As noted previously, the <smil> element is simply the outer container of the markup “sandwich.” The <head> and <body> sit within this outer container. In SMIL 1.0, the <smil> element had an optional attribute of id. SMIL 2.0 added the attributes class, xml:lang, title, and xmlns.

In both QuickTime and RealPlayer, a basic use of the <smil> element would take the following form in a text document:

```
<smil>
...presentation <head> and <body> elements and their contents would be
  inserted here...
</smil>
```

Since Internet Explorer is an Integration Set implementation, the <smil> element is not used. Instead, SMIL timing characteristics are identified with the xmlns attribute as covered later in this chapter. A sample would look something like this:

```
<html xmlns:t="urn:schemas-microsoft-com:time">
```

Element: <head>

- Version Compatibility: SMIL 1.0, SMIL 2.0
- Possible Attributes: id, class, xml:lang, title

- Parent Element: <smil>
- Child Elements (in SMIL 1.0): <layout>, <meta>, <switch>

The <head> element provides information that is not related to the timing of the media elements (called nontemporal information). Some of the choices available include providing metainformation to describe the presentation and media elements involved; layout capabilities, which determine where on the screen the media will appear; and matching your content and delivery to the viewer's situation. In the SMIL 1.0 specification, the <meta>, <layout>, and <switch> elements are established as children of the <head> element. Although their location may be the same in SMIL 2.0, the newer specification breaks those elements off to their own modules.

With the exception of the namespace (xmlns) attribute, the <head> element offers the same attributes as the <smil> element. You will be more likely, however, to attach a title to your presentation in the <head> element than in <smil>. For more information on controlling the <layout> of your presentation, visit Chapter 6, "Placing It on the Page: The Layout Modules." The Metainformation module (<meta>) is covered in Chapter 5, "Data About Data: The Metainformation Module," and controlling the nature of your presentation with <switch> is covered in Chapter 12, "Optimizing Content Delivery: Content Control Modules."

As with the <smil> element usage in both QuickTime and RealPlayer, a basic use of <head> would take the following form in a text document:

```
<smil>
  <head>
    ...<meta> and <layout> element information is an example of what
      would typically be inserted here...
  </head>
  <body>
    ...This is where the markup for the presentation goes...
  </body>
</smil>
```

In versions of Internet Explorer that support HTML+TIME, the <head> element is really the HTML <head> element. This would carry the link to the time behaviors supported by HTML+TIME. In an HTML text document, the form would look like this:

```
<html xmlns:t="urn:schemas-microsoft-com:time">
  <head>
```

```
<style>
    .time {behavior:url(#default#time2);}
</style>
<?IMPORT namespace="t" implementation="#default#time2">
...additional <html> head elements as needed by Web page...
</head>
</html>
```

Element: <body>

- Version Compatibility: SMIL 1.0, SMIL 2.0
- Possible Attributes: id, class, xml:lang, title
- Parent Element: <smil>
- Child Elements (in SMIL 1.0): <a>, <animation>, <audio>, , <par>, <ref>, <seq>, <switch>, <text>, <textstream>, <video>

The <body> element contains content and pathways to your content, as well as all the information related to the timing of the presentation, including start and stop times and sequencing.

With the exception of the namespace (xmlns) attribute, the <body> element offers the same attributes as the <smil> and <head> elements. You will be more likely, however, to attach an id to your presentation elements in the <body> element than in <smil> or <head>.

In SMIL 1.0, the specification offers the <a>, <animation>, <audio>, , <par>, <ref>, <seq>, <switch>, <text>, <textstream>, and <video> elements as children of the <body> element. Again, although their location may be the same in SMIL 2.0, the newer specification breaks those elements off to their own modules. For more information on controlling the flow of your presentation with <switch>, see Chapter 12. Chapter 8, "Bring on the Media: The Media Object Modules," covers the <animation>, <audio>, , <text>, <textstream>, <ref>, and <video> elements used to contain and display your media. If you want to link with the <a> anchor element, visit Chapter 11, "Making It Interactive: The Linking Modules." Finally, the critical <par> parallel and <seq> sequential timing elements are covered in Chapter 7, "Grouping Your Content: The Timing and Synchronization Module."

Both QuickTime and RealPlayer make basic use of <body> with the following form in a text document:

```
<smil>
  <head>
    ...This is where the markup for information about your content
    goes...
  </head>
  <body>
    ...presentation content instructions would be inserted here...
  </body>
</smil>
```

As with the `<head>` element in Internet Explorer, `<body>` in IE is really the `<body>` of the HTML document itself. Since an HTML `<body>` element contains the content of the presentation, the end goal is the same as a traditional SMIL `<body>` element.

The Structure Module Attributes

The `<smil>`, `<head>`, and `<body>` elements all make use of the following attributes. (The only exception is `xmlns`, or the namespace declaration attribute, which is used exclusively by the `<smil>` element.)

Attribute: `id`

- Version Compatibility: SMIL 1.0, SMIL 2.0
- Possible Values: Valid XML identifier

The `id` attribute attaches a unique identifier to the element in question. This is similar to the `name` attribute in traditional HTML. Assigning a value to `id` allows you to reference that element later with another attribute in SMIL.

Attribute: `class`

- Version Compatibility: SMIL 2.0
- Possible Values: Valid class name or class names separated by spaces

This attribute is used to assign a class name or a set of class names to an element. It is almost identical to `id` but `class` allows any number of names to be assigned to an element or group of elements.

Attribute: xml:lang

- Version Compatibility: SMIL 2.0
- Possible Values: Valid language code (with options from Ami to English to Klingon to Xiang)

As covered in the XML specification, this attribute identifies the language used in the presentation content and by the relevant attributes in the markup.

Attribute: title

- Version Compatibility: SMIL 2.0
- Possible Values: An alphanumeric text string

This is intended to let the user know something about the element to which it is attached. Implementation depends on its use; the information might be displayed in a title bar or seen as a mouse is rolled over an object, for example.

Attribute: xmlns

- Version Compatibility: SMIL 2.0
- Possible Values: A declared namespace

You will find this is the attribute attached to only one element: `<smil>`. `xmlns` declares the appropriate namespace in use in the SMIL document. This attribute allows for a check that compares the SMIL used in your document against standards that are linked to the namespace attribute. As you would expect considering its function, it is found only as an attribute of `<smil>`.

QuickTime makes use of the `xmlns` attribute to reference to Apple's own QuickTime extensions to SMIL:

```
<smil xmlns:qt="http://www.apple.com/quicktime/resources/smilextensions">
```

RealPlayer makes a direct reference to the W3C implementation of SMIL 2.0 with:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/CR/">
```

Like QuickTime, RealPlayer also offers RealNetworks's own extensions to SMIL. The markup form is the same. So, for example, the markup to reference Alpha Channel and Resizing namespaces would look like:

```
<smil xmlns="http://www.w3.org/2001/SMIL20/Language"
  xmlns:rn="http://features.real.com/2001/SMIL20/Extensions">
```

Probably the most involved use of the namespace attribute is in Internet Explorer browsers that support their HTML+TIME implementation of SMIL. You must declare the namespace, import the tag definitions, and then provide a means to associate elements with their time2 behavior. To cover the bases, simply use the following code in your HTML document:

```
<html xmlns:t="urn:schemas-microsoft-com:time">
  <head>
    <style>
      .time {behavior:url(#default#time2);}
    </style>
    <?IMPORT namespace="t" implementation="#default#time2">

  </head>
</html>
```

That use of the `<style>` element is the key that provides a critical link so that your HTML elements can pick up the SMIL behaviors.

Summary

The use of the `<smil>`, `<head>`, and `<body>` elements in SMIL 1.0 and 2.0 is remarkably similar. Additional attributes for these elements in 2.0 offer more flexibility and options. The implementations vary somewhat between the various players and browsers. With Internet Explorer especially, Structure is actually handled by the HTML document itself, so SMIL capabilities are linked to the HTML. But despite all the differences among the implementations, the goal is the same. Structure establishes the framework in which your presentation will sit.



Grouping Your Content: The Timing and Synchronization Module

- Conformance Profiles and the Timing and Synchronization Module
- The Timing and Synchronization Module Elements
- The Timing and Synchronization Module Attributes
- The Timing and Synchronization Module Summarized

At the heart of the Synchronized Multimedia Integration Language lies the Timing and Synchronization module. Orchestrating the timing of various media elements in the presentation is, after all, what makes SMIL different from traditional Web media.

SMIL 1.0 introduced the critical `<seq>` and `<par>` elements for the sequential and parallel playback of media. SMIL 2.0 adds to the capabilities by offering an `<excl>` element for exclusive playback.

Despite how simple it is to implement, Timing and Synchronization is the heftiest and most important module in SMIL. From here, chaos is converted to order and the form of the presentation takes shape.

Conformance Profiles and the Timing and Synchronization Module

As a basic building block in SMIL, you will find the Timing and Synchronization module used both for SMIL Host Language and Integration Set Language Conformance. SMIL Host Language Conformance will lean toward using the `<seq>`, `<par>`, and `<excl>` elements. Because Integration Set Conformance involves incorporating the language into other XML languages, the `timeContainer` attributes with their `seq`, `par`, and `excl` values become important in that arena.

The Timing and Synchronization Module Elements

- Module: Timing and Synchronization
- Conformance: Required for SMIL Host Language and SMIL Integration Set
- Profile Compatibility: SMIL 1.0, SMIL 2.0, SMIL 2.0 Basic, XHTML+SMIL
- Possible Elements: `<seq>`, `<par>`, `<excl>`

In SMIL 1.0, the `<seq>` and `<par>` tags of SMIL Timing and Synchronization were two of the most notable elements of the language. `<seq>` and `<par>` act as time containers, allowing you to organize and group your content. `<seq>` plays contained elements in sequence. `<par>` plays contained elements in parallel. By listing and nesting these two elements in various ways, you can build almost any presentation.

SMIL 2.0 gains the `<excl>` element. `<excl>` plays contained elements exclusively—that is, one grouping at a time. But in doing so, it allows the SMIL presentation to jump and branch for a great deal of interactivity. Supporting `<excl>` is the `<priorityClass>` element, which allows the SMIL author to control the playback behavior of the objects that the presentation has not jumped or branched to.

Element: `<seq>`

- Version Compatibility: SMIL 1.0, SMIL 2.0
- Possible Attributes: `begin`, `dur`, `end`, `fill`, `fillDefault`, `higher`, `lower`, `max`, `min`, `pauseDisplay`, `peers`, `repeat`, `repeatCount`, `repeatDur`,

```
restart, restartDefault, syncBehavior, syncBehaviorDefault,  
syncMaster, syncTolerance, syncToleranceDefault, timeAction
```

The `<seq>` element was one of the initial timing building blocks of SMIL. Along with its `<par>` and `<excl>` siblings, `<seq>` can be called a “time container.” It contains a child set of elements and specifies how they will relate to each other along a timeline. It plays each child element sequentially, as it is listed. Let’s suppose you have two media objects referenced between a pair of `<seq>` tags. The first media object would play in its entirety, and then the second media object would play. For example, you could use `<seq>` to play a sequence of images in a slide show presentation.

Using `<seq>` is not difficult in most implementations. QuickTime uses `<seq>` in the following traditional manner to play one clip followed by another:

```
* * *  
<seq>  
  <audio src="myAudio.mp3" />  
  <video src="myVideo.mov" />  
</seq>  
* * *
```

Other than possibly different media types, a sample of `<seq>` RealPlayer will look almost identical:

```
* * *  
<seq>  
  <audio src="myAudio.rm" />  
  <video src="myVideo.rm" />  
</seq>  
* * *
```

By contrast, Internet Explorer takes on a decidedly different look when implementing `<seq>`:

```
* * *  
<t:seq>  
  <div class="time" dur="2">First line of text</div>  
  <div class="time" dur="2">Second line of text</div>  
</t:seq>  
* * *
```

This difference in code is caused by Internet Explorer's Integration Set approach, in which the SMIL markup is mixed in with the HTML of a Web page.

The `<seq>` element supports all element timing attributes except for two: `endsync` and `timeContainer`. The `timeContainer` attribute can't be added to a `<seq>` because this attribute is a type of time container itself.

But you can use `timeContainer` to attach `<seq>` (or `<par>` and `<excl>`) characteristics to any XML component. This enables you to bring SMIL to languages such as XHTML. So, as seen in Microsoft Internet Explorer, a `<seq>` using `timeContainer` would look like this:

```
* * *
<span class="time" timeContainer="seq">
  
  
  
</span>
* * *
```

Element: `<par>`

- Version Compatibility: SMIL 1.0, SMIL 2.0
- Possible Attributes: `begin`, `dur`, `end`, `endsync`, `fill`, `fillDefault`, `higher`, `lower`, `max`, `min`, `pauseDisplay`, `peers`, `repeat`, `repeatCount`, `repeatDur`, `restart`, `restartDefault`, `syncBehavior`, `syncBehaviorDefault`, `syncMaster`, `syncTolerance`, `syncToleranceDefault`, `timeAction`

Another of the initial timing building blocks of SMIL, the `<par>` element plays child elements in parallel. If you had two media objects referenced between a pair of `<par>` tags, the two media objects would play at the same time. For example, you could use `<par>` to play an animation presentation and its audio soundtrack simultaneously.

Like `<seq>`, putting `<par>` to work is not difficult. QuickTime uses `<par>` in the following way:

```
* * *
<par>
  <audio src="myAudio.mp3" />
  <video src="myVideo.mov" />
</par>
* * *
```

QuickTime will play both media objects in parallel. As with <seq>, RealPlayer is almost identical in its use of <par>:

```
* * *
<par>
  <audio src="myAudio.rm" />
  <video src="myVideo.rm" />
</par>
* * *
```

As expected, the SMIL Integration Set approach of Microsoft Internet Explorer provides a different look:

```
* * *
<t:par>
  <div class="time" dur="10">First line of text</div>
  <div class="time" dur="10">Second line of text</div>
</t:par>
* * *
```

In this example, two lines of text will appear on the screen at the same time and disappear 10 seconds later.

The <par> element supports all element timing attributes except for one. You cannot apply a timeContainer attribute to a <par>, because this attribute is a type of time container itself.

But, as with <seq>, you can use timeContainer to attach <par> characteristics to any XML component. This allows you to bring SMIL to languages such as XHTML. In Internet Explorer, using timeContainer for a <par> would look like this, which makes three graphics play on the screen at the same time:

```
* * *
<span class="time" timeContainer="par">
  
  
  
</span>
* * *
```

Although you can use <par> to have multiple media elements running at the same time, you might not want to. What I mean by this is that <par> makes it very easy to overload the available bandwidth. An audio file running here, a movie running at the same time over there, and suddenly you have a presentation that may use more bandwidth than you have available. When using <par>, keep a tight eye on the total bandwidth that each media object consumes.

Throughout the SMIL modules, the language offers a number of elements and attributes that enable you to determine what type of content a viewer receives or tweak the bandwidth-influenced behaviors of that content. You have a number of options to help you manage getting that fat content of yours down a narrow pipe. As you read the specification, keep an eye out for these features and use them whenever helpful.

Element: <excl>

- Version Compatibility: SMIL 2.0
- Possible Attributes: begin, dur, end, endsync, fill, fillDefault, higher, lower, max, min, pauseDisplay, peers, repeat, repeatCount, repeatDur, restart, restartDefault, syncBehavior, syncBehaviorDefault, syncMaster, syncTolerance, syncToleranceDefault, timeAction

New to SMIL 2.0, <excl> offers a unique twist to the Timing and Synchronization elements. <excl> allows only one child element to be played at a time. That might sound like the <seq> element, but the <excl> element imposes no sequential order. Its only concern is that only one element plays at a time. If an element begins to play while another element is playing, the new element takes charge and the old element is paused or stopped.

How might this be useful? Suppose you have supplementary material that supports your main presentation. By using <excl>, your viewer could pause the main presentation, branch off to the supplementary material, and then return to pick up where the user left off in the main presentation. In essence, <excl> becomes the new interactivity powerhouse in SMIL 2.0. Some interactivity was possible with creative use of SMIL 1.0, but SMIL 2.0 makes it easier.

At its most basic, using <excl> in RealPlayer might look something like this:

```
* * *
<excl>
  
  
</excl>
* * *
```

In this example, the second image actually plays (exclusively) before the first image because of the begin timing. But this is an overly simplistic sample. A SMIL 2.0 player such as RealPlayer can do so much more. Consider the follow-

ing example from Internet Explorer. It shows a more complex, but still simple, two-image gallery linked to two buttons. <excl> provides the interactive power:

```
. . .
<button id="first">First Image</button>
<button id="second">Second Image</button>

<t:excl dur="indefinite">

  <t:par begin="first.click">
    
    <div class="time" dur="5" timeAction="display">The first
image</div>
  </t:par>

  <t:par begin="second.click">
    
    <div class="time" dur="5" timeAction="display">Sunset at an
African Beach</div>
  </t:par>
</t:excl>
. . .
```

With each click of a button on the screen, a corresponding parallel element begins displaying an image and text. This is because each parallel element within the exclusive grouping has a *begin* value linked to a click of a named button. A lot is going on in this sample, but it shows how a <par> element can be called with a button click if that parallel element is wrapped in a <excl>. With <excl>, SMIL 2.0 gains powerful new interactivity.

The <excl> element can work closely with the <priorityClass> element from the Content Control modules (see Chapter 12, "Optimizing Content Delivery: The Content Control Modules") to control interactivity. The <priorityClass> element groups child elements together and enables you to control the pause, resume, and interruption characteristics of those children as they become active in the timeline.

The <excl> element supports all element timing attributes except for the *timeContainer* attribute; this attribute is a type of time container itself.

But you can use *timeContainer* to attach <excl> characteristics to any XML component. This allows you to bring SMIL to languages such as XHTML. A *timeContainer* for <excl> in Internet Explorer would look like something like this:

```
* * *
<span class="time" timeContainer="par">
  
  
  
</span>
* * *
```

A user clicking a button (created elsewhere in the markup) could choose one of three potential images.

Element: <priorityClass>

- Version Compatibility: SMIL 2.0
- Possible Attributes: peers, lower, higher

<priorityClass> adds power to the <excl> element by enabling the SMIL author to control the specific behavior of supporting objects when a given object is chosen. This control can be forced upon a peer, a lower element, or a higher element. Typical behaviors offered by these attributes allow the author to pause or stop the playback of supporting elements when a new element is chosen. In some uses of <priorityClass>, the new element choice will be ignored until the previous element finishes its playback.

<priorityClass> is put to work by making it a child of an existing <excl> element. For example, in RealPlayer, <priorityClass> and a peers attribute can be used to pause one clip when another begins playing:

```
* * *
<excl>
  <priorityClass peers="pause">
    <video id="first" src="first.rm" />
    <video id="second" src="second.rm" />
  </priorityClass>
</excl>
* * *
```

Because the video clips are peers of each other, running one clip will pause the other.

Using <priorityClass> within Internet Explorer is pretty much the same. In the following example, activation of the second line of text forces the first one to pause in the timeline:

```
* * *
<t:excl>
  <t:priorityClass peers="pause">
    <span class="time" begin="0" dur="15">
      First line of text displayed.
    </span>
    <span class="time" begin="5" dur="10">
      Second line of text is displayed, interrupting the first.
    </span>
  </t:priorityClass>
</t:excl>
* * *
```

The Timing and Synchronization Module Attributes

If `<seq>`, `<par>`, and `<excl>` elements group the elements of your content, the Timing and Synchronization attributes provide much of the control over when those groups play. Much of this work is done by the critical `begin` and `end` attributes. However, you will also find attributes that allow you to tweak your presentation for varying bandwidth conditions. Finally, `timeContainer` and `timeAction` give you attributes for use in other environments, allowing new SMIL power for languages such as XHTML.

Attribute: `begin` and `end`

- Version Compatibility: SMIL 1.0, SMIL 2.0
- Possible Values: See the following for basic descriptions. `begin` and `end` support a wide range of options, including `begin-value-list`, `offset-value`, `syncbase-value`, `event-value`, `repeat-value`, `accesskey-value`, `media-marker-value`, `wallclock-sync-value`, `indefinite`

If the `<seq>`, `<par>`, and `<excl>` elements are the building blocks of SMIL timing, the `begin`, `dur`, and `end` attributes are the workers helping to put it all together. Because the values of `begin` and `end` are identical, I have combined the two attributes in this one section.

To those new to SMIL, all the possible values I list here will seem a bit intimidating. Don't let the options scare you. If you're new to SMIL, a `begin` can be as easy as saying `begin="2s"`—if you want to begin your element at 2 seconds into the timeline. But `begin` and `end` attributes can also mean so much more. What if you wanted an element to begin only when another element ends? Or what if you

want an element to begin when a mouse click occurs? These attributes offer just that kind of power.

My coverage of begin and end values starts simply enough: with basic SMIL 1.0 legal clock values. These might be reflected in full clock, partial clock, or timecount values. An example of a full clock value might be 01:15:03 (as in `begin="01:15:03"`); the element will begin playing at 1 hour, 15 minutes, and 3 seconds. With a partial clock value, unneeded time components such as hours are hidden. For example, `begin="00:30.25"` will cause the element to play at 30.25 seconds. For many SMIL authors, timecount values are probably the most convenient to use. An example would be an `end="25s"` declaration. This would end the element at 25 seconds. In their abbreviated form, timecounts are easy to figure out quickly and enter in as a value in an attribute. Most people understand 25 seconds as 25s more easily than 00:00:25.0.

Implementations can vary in preferred ways of representing time values. That is not to say that one player cannot understand the time value used by another. But it is important to keep an eye on these issues. For example, a begin in QuickTime would look like this:

```
<video src="myMovie.mov" begin="10sec" />
```

In RealPlayer, a begin attribute would look like the following:

```
<video src="myMovie.rm" begin="10s" />
```

Finally, an Internet Explorer begin attribute would appear as follows:

```
<div class="time" begin="10">My Text</div>
```

You can also use offset values with begin and end. An offset is simply the negative or positive space off of the given time. Thus, a `begin="+5s"` means that the element should begin 5 seconds after the parent element begins. In a grouping with other media objects, this might mean your audio starts 5 seconds after the first image appears.

With SMIL 2.0, more interactivity enters the fray with the use of the `accesskey` value for begin and end. This looks for the user to press a specific key.

`end="accesskey(y)"` tells the player or browser to end the element when the "y" key is pressed.

Both SMIL 1.0 and 2.0 support extensive use of `id` to relate begin and end times to identified elements. Before we get too far, some review of `id` is in order. `id`, an attribute of the Structure module, can be attached to any element in a SMIL document. As covered earlier, it requires a "valid XML identifier," but that can really

mean nothing more than a short, text-based name. Therefore, attaching `id="1"`, or `id="first"`, or even `id="firstSection"` to a time container or media object will identify that element with that name. After something is named, it can be called by name for use in other elements and attributes.

The use of `id` has changed a little from SMIL 1.0 to SMIL 2.0, but the end goal is basically the same. Let's take a quick look at the differences. In the following generic example of SMIL 1.0, an `id` of "fred" is given to the first clip in the `<par>` block:

```
<par>
  <audio src="myaudio" id="fred" />
  <animation src="myanimation" begin="id(fred)(5s)" />
</par>
```

In this example of SMIL 1.0, the `<animation>` element references the `<audio>` element called "fred" in its `begin` value and tells "myanimation" to begin 5 seconds after "fred" begins. In SMIL 2.0, this reference might be rewritten to look like this:

```
<par>
  <audio src="myaudio" id="fred" />
  <animation src="myanimation" begin="fred.begin+5s" />
</par>
```

The end result is the same, but the first example is deprecated in SMIL 2.0. That means that the SMIL 1.0 approach can be used, but the W3C does not recommend it.

So what can `id` do for you? You are probably beginning to get just a taste already. Suppose that you want to synchronize the actions of one element to the status of another. `begin="fred.begin"` sounds redundant, but it really means that the current element will begin when the element identified as "fred" begins. You also might use `begin="fred.end"` to start the new active element in motion after "fred" ends. And I'm just getting started with the possibilities. Suppose you wanted the current element to `end="fred.end+5s"`. This would end the element 5 seconds after "fred" ends; or why not try `end="fred.begin-5s"` to end the element 5 seconds before "fred" begins. Notice that I'm using the same values for both the `begin` and `end` attributes. Likewise, you can mix and match in your work as well.

An example of this kind of timing with an Internet Explorer begin attribute would appear as follows:

```
<div class="time" begin="jane.begin+10">My Text</div>
```

Now that you have some id basics down, let's look for more options. SMIL 2.0 supports additional teaming of id with media-marker values or wallclock-sync values. If you use a media type that supports markers, you might embed the value "sally" somewhere within the timeline of that media object. Now by identifying that media object as "fred", you have opened the door to accessing that marker. `begin="fred.marker(sally)"` will begin the current element when the media object called "fred" hits its internal marker labeled as "sally".

Wallclock values allow the begin or end attributes to make the connection with the outside world. Several time formats represent global and local time approaches. For example, `end="wallclock ( 2002-01-01T12:00:00 )"` would force the element to end on January 1, 2002 at 12 p.m. in the U.S. Pacific time zone. `wallclock-sync` values create the path for live SMIL-driven events.

One of the most important aspects of SMIL 2.0 id use is in connecting begin and end to events. What is an event? An event is an occurrence that the computer can recognize and acknowledge. It could be the playing of a media clip, or with a user action event, it could be the click of a mouse.

This provides powerful interactivity without the need for using a scripting language. When implemented in a Web browser that supports SMIL, the result can feel a lot like JavaScript. In a Web multimedia SMIL player, event-driven use of begin and end makes SMIL 2.0 a vast improvement over SMIL 1.0. Event-driven begin and end is especially handy when used with the `<excl>` element.

SMIL 2.0 defines three events: `beginEvent`, `endEvent`, and `repeat`. `beginEvent` occurs when the local element timeline begins to play. `endEvent` occurs when the active element ends in the local timeline. `repeat` looks for the local timeline repeating. Coupled with other events that begin or end other elements in the timeline, `beginEvent`, `endEvent`, and `repeat` can kick in to move the presentation on to its next step.

The `repeat` event offers additional hooks to get at what is happening in the timeline. The form uses the id, followed by a dot and the keyword `repeat`, followed by an iteration (or loop) value in parentheses. Therefore, `end="fred.repeat(2)"` means that the element will end when "fred" begins its second iteration (loop).

Additional action events are defined by the environment in which the player or browser exists. Click your mouse and you have just created an occurrence that the computer can recognize and acknowledge. This could be described as a “click” action. Depending on the environment, other occurrences could be rolling the mouse point over an object (a “mouseover”) or even a dragging one object on top of another.

An example of an event-driven use of `begin` is `begin="fred.click"` or `begin="fred.activateEvent"`. The first example is from HTML+TIME support in Internet Explorer. The second is from SMIL 2.0 support in RealPlayer. Both use a familiar dot syntax approach in which the event is tied to an object that is labeled with an `id` attribute. In the case of both examples, this happens to be some type of media object to which the author has added the `id="fred"` attribute. In the case of Internet Explorer, the user event of clicking the mouse is known as a `click`. Clicking the mouse in RealPlayer is known as an `activateEvent`. Despite the minor syntax differences, both look for a mouse click happening to “fred” and force the element to begin.

As with other value options with `begin` and `end`, events can also pick up clock-value offsets. These are times added to or subtracted from the timing of the event. Again, we see the familiar dot syntax. `begin="fred.click+2s"` will delay the `begin` by 2 seconds when “fred” is clicked. `end="fred.repeat(2)-5s"` will likewise end the element 5 seconds before “fred” begins its second iteration (loop) in a repeat.

One additional power of `begin` and `end` that might not be immediately obvious is that you can specify multiple `begin` and `end` times, although `begin="5s"` might delay the element by 5 seconds, you could also double up the work of that statement by converting it to `begin="5s; fred.click"`. A semicolon separates the two (or more) events. In this example, the element will begin after 5 seconds and/or after “fred” is clicked.

Both `begin` and `end` support a set value of `indefinite`. Depending on whether the provided attribute is `begin` or `end`, using `indefinite` means that a subset of methods will determine the beginning or ending value. With `begin`, the methods `beginElement()` and `beginElementAt()` can begin the active duration of an element. This will also begin the active duration of an element with a hyperlink targeted to the element. With `end`, the methods `endElement()` and `endElementAt()` can determine the active duration of the element. How can this be used? In HTML+TIME implementations of Internet Explorer for example, an

`endElement()` method linked to a Web page button can be used to shut down a continuously repeating object. Following is a HTML+TIME markup sample that shows just that in action:

```
* * *
<t:excl id="myText" repeatDur="indefinite">
  <div class="time" begin="0" dur="4">It was a dark and stormy
  night.</div>
  <div class="time" begin="4" dur="5">The Captain turned to me and asked me
  to tell a story. And so I began. . .</div>
</t:excl>
<br>
<button onclick="myText.endElement()">Stop It!</button>
* * *
```

In this example, my attempt at a short story will loop indefinitely. But with the `endElement()` method attached to "myText" in the `onClick` statement in the last line of the code snippet, I have a way out. I can click the button and make it all end.

Attribute: dur

- Version Compatibility: SMIL 1.0, SMIL 2.0
- Possible Values: A legal clock value greater than or equal to 0, *media*, or *indefinite*

After you have a beginning for an element established with `begin`, it is time to give the element a duration. SMIL incorporates two types of duration terminology. A *simple duration* is the basic presentation length of a single element. An *active duration* is the sum of simple duration and any repeated events of that simple duration.

The `dur` attribute establishes the simple duration of a media element. It supports clock values greater than 0. You can also use the value *media* to set the duration of a media object to the inherent duration of the media itself. A value of *indefinite* sets the value at an indefinite duration in time.

How the element is impacted by the `dur` value is based on the nature of the element itself. For media that have built-in durations (such as video or audio), specifying a shorter duration would potentially chop short that media object. Specifying a longer than normal duration might mean that the viewer is left with "dead air." This would not be true for an object that has no predefined duration.

For example, a slide show of images would follow the lead of whatever durations you specify in your markup.

In QuickTime, a simple duration would use the following form:

```

```

Similarly, RealPlayer appears almost identical in use

```

```

In Internet Explorer, a duration value would appear as

```
<div class="time" dur="10">My text</div>
```

Attribute: min

- Version Compatibility: SMIL 2.0
- Possible Values: A legal clock value greater than or equal to 0 or media

As stated earlier, within SMIL timing, the author makes use of two types of duration: simple and active. Playing an element through from beginning to end would be the simple duration. Playing that element through three times would create an active duration.

The `min` attribute defines what the minimum active duration of an element will be. If a clock value is given, the element will play at least that much material before moving on. Even if the user makes an interactive choice, the `min` attribute will lock that user into the presentation until that minimum active duration is reached. If you are using a media element, the `media` value locks the active duration down to the duration of that media element.

Attribute: max

- Version Compatibility: SMIL 2.0
- Possible Values: A legal clock value greater than or equal to 0, media, or `indefinite`

The `max` attribute picks up as the opposite of what the previously discussed `min` attribute sets out to do. In this case, the element cannot play longer than the specified clock value. If `media` is the set value for a media element, the element cannot play longer than the media itself. If `indefinite` is specified, no limitation on the element is made.

Attribute: endsync

- Version Compatibility: SMIL 1.0, SMIL 2.0
- Possible Values: first, last, all, media, or a legal XML Id value

endsync first appeared in SMIL 1.0. This attribute allows the author to end the children of <par> or <excl> elements on the basis of one of the children reaching the end of its duration. This is handy for objects that do not have a readily defined or known duration.

In SMIL 1.0, endsync offers the first, last, and Id-value options. Using a first value tells the element to end its group along with the very first child to reach the end of its duration. Using last means that the element group will end when the very last child reaches its complete duration. If you think about it, this essentially means that everything will end in its own time. So you will find that last is the default value for endsync. You can also endsync with a legal XML Id value. For example, if endsync="id(fred)", the group will end when the clip labeled as fred ends. The Id-value approach has been updated slightly in SMIL 2.0, although most authors will barely notice the difference in the syntax between the two uses.

Newer implementations of SMIL in QuickTime support the SMIL 1.0 approach to endsync. In that environment, you might see endsync used with first, last, or as shown below, id:

```
* * *
<par endsync="id(myAudio)">
  <audio id="myAudio" src="narration.mov" />
  <video src="theVideo.mov" />
</par>
* * *
```

In SMIL 2.0, the additional values of all and media are added to the mix. all means that the grouping of children ends when all of them have reached the end of their durations. The value of media is meant for media objects. Using media as an endsync value means that the time container itself will end with the end of the media object in that time container.

Attribute: repeat

- Version Compatibility: SMIL 1.0, SMIL 2.0
- Possible Values: Integers greater than 0 and indefinite

SMIL 1.0 introduced the `repeat` attribute to allow for an element to repeat a specified number of times. The value for the number of iterations of the repeat can be specified as an integer greater than 0. For continuously repeating elements, a value of `indefinite` can also be used.

SMIL 2.0 lists `repeat` but deprecates the usage in favor of the more flexible `repeatCount` and `repeatDur` attributes.

Currently, Apple QuickTime prefers the `repeat` form of the language:

```
<video src="myMovie.mov" repeat="2" />
```

Attribute: `repeatCount`

- Version Compatibility: SMIL 2.0
- Possible Values: A floating-point numeric value or `indefinite`

`repeatCount` specifies how many iterations (or loops) of the simple duration of the element.

In the following sample of SMIL from RealPlayer, the video clip will repeat 3 times:

```
<video src="myMovie.rm" repeatCount="3" />
```

Using a fractional value will cause a fractional portion of the duration. In the following example of a fractional value used in Internet Explorer, the sequence of images within the time container will display two and a half times:

```
. . .
<span class="time" timeContainer="seq" repeatCount="2.5">
  
  
  
</span>
. . .
```

If you want the element to repeat continuously, use `indefinite` as the value.

Attribute: `repeatDur`

- Version Compatibility: SMIL 2.0
- Possible Values: A legal clock value or `indefinite`

While `repeatCount` controls how many iterations or partial iterations of the simple duration of an element appear in the timeline, `repeatDur` takes a time-based

approach. In other words, you might ask yourself how much time you want to fill with the element repeating. Then, by your specifying a clock value, the element will repeat as many times as necessary to fill that amount of time. Like `repeatCount`, using `indefinite` as a value will cause the element to repeat continuously.

In this sample of SMIL from RealPlayer, the video clip will repeat for a duration of 2 minutes:

```
<video src="myMovie.rm" repeatDur="2min" />
```

Attribute: fill

- Version Compatibility: SMIL 1.0, SMIL 2.0
- Possible Values: `remove`, `freeze`, `hold`, `transition`, `auto`, `default`

`fill` allows Web multimedia authors to provide a smoother presentation by accounting for different duration times in various elements on a timeline. The `fill` attribute first appeared in SMIL 1.0. Possible values in SMIL 1.0 include `remove` and `freeze`. If the `remove` value is used, the element will disappear after its simple duration is finished. If the `freeze` value is used, the element will freeze when its simple duration is finished. For example, a video clip might freeze on the last frame of the video.

SMIL 2.0 adds the additional attribute values of `hold`, `transition`, `auto`, and `default`. `hold` has the same effect as `freeze` with one important difference. Using `hold` means that the element will stay frozen only until the parent group of elements is finished. With `transition`, the element will stay frozen at the end of its duration until a transition effect is finished. The `auto` value creates a variable result depending on other conditions. If you have specified a value or values for an attribute such as `dur`, `end`, `repeatCount`, or `repeatDur`, the `fill` value will automatically set itself to `freeze`. But if you have not specified a value for any of those attributes, the `fill` value will automatically behave as a `remove`.

SMIL 2.0 also offers a default attribute value. This value forces the element to take its lead from the `fillDefault` value specified in the parent element.

In the following sample from RealPlayer, `fill` keeps text on the screen even as the movie clip ends:

```
* * *
<par>
  <video src="myMovie.rm" region="movie" />
  <textstream src="myText.rt" region="credits" fill="freeze" />
</par>
* * *
```

Attribute: fillDefault

- Version Compatibility: SMIL 2.0
- Possible Values: remove, freeze, hold, transition, auto, inherit

SMIL 2.0 adds the `fillDefault` attribute to the Timing and Synchronization bag of tricks. `fillDefault` offers the same `remove`, `freeze`, `hold`, `transition`, and `auto` values of `fill`. With `fillDefault`, an entire block of elements can be set with one attribute value. `remove` causes the element to disappear after its simple duration has ended. `freeze` causes the element to end as frozen after it has completed its duration. `hold` keeps the element frozen until the other elements in the group are finished. `transition` waits until a scheduled transition is finished before removing the element. Like the `fill` attribute, `auto` determines a `freeze` or `remove` behavior depending on whether a `dur`, `end`, `repeatCount`, or `repeatDur` value is specified.

`fillDefault` adds one other attribute value to the set offered by `fill`. If a value of `inherit` is used, `fillDefault` will look to the `fillDefault` of a parent element to determine what its value should be.

In the following sample from `RealPlayer`, `fill` is set to "freeze" across all the objects in the time container:

```
...
<par fillDefault="freeze">
  <video src="myMovie.rm" region="movie" />
  <textstream src="myText.rt" region="credits" />
</par>
...
```

Attribute: higher

- Version Compatibility: SMIL 2.0
- Possible Values: stop, pause

The `higher` attribute extends the previously listed `peers` attribute by allowing elements with a higher priority to affect the children elements. With `stop` specified, after a higher priority element becomes active, it forces the child elements to stop in the timeline. `pause` forces the child elements to pause when the higher priority element becomes active. When the higher priority element is finished, the child elements resume.

The following sample shows how higher can be used in RealPlayer:

```
* * *
<excl>
  <priorityClass id="first">
    <video id="clip1" src="first.rm" />
    <video id="clip2" src="second.rm" />
  </priorityClass>
  <priorityClass id="second" higher="pause">
    <video id="clip3" src="third.rm" />
  </priorityClass>
</excl>
* * *
```

The first block of video clips is a higher class than the second block containing the third clip. If a clip in the first block begins to play while clip3 is playing, higher="pause" means that clip3 will immediately pause.

The markup for higher in Internet has the same result:

```
* * *
<t:excl>
  <t:priorityClass>
    <span id="first" class="time" begin="5" dur="15">
      This line of text interrupts the following line.
    </span>
  </t:priorityClass>
  <t:priorityClass higher="pause">
    <span id="second" class="time" begin="0" dur="10">
      This line of text displays first.
    </span>
  </t:priorityClass>
</t:excl>
* * *
```

Here, the second line of text will display at the beginning. But at 5 seconds into the timeline, the higher class will pause the second line of text.

Attribute: lower

- Version Compatibility: SMIL 2.0
- Possible Values: defer, never

Much like the previously listed higher attribute, the lower attribute enables the author to control the child elements, depending on how the lower priority elements behave. If a lower-priority element becomes active in the timeline, a value

of defer will force that element to wait until the active child elements are completed. If the never value is used, the lower priority element will never become active if the child elements are currently active.

The following sample from RealNetworks shows lower in action:

```
...
<excl>
  <priorityClass id="first">
    <video id="clip1" src="first.rm" lower="never" />
  </priorityClass>
  <priorityClass id="second">
    <video id="clip2" src="second.rm" />
  </priorityClass>
</excl>
...
```

In this example, if clip2 attempts to play, it could potentially interrupt clip1. lower="never" tells the player not to let that happen. clip2 cannot play until clip1 is done.

Internet Explorer markup follows a similar approach:

```
...
<t:excl>
  <t:priorityClass lower="never">
    <span id="first" class="time" begin="0" dur="15">
      This line of text interrupts the following line.
    </span>
  </t:priorityClass>
  <t:priorityClass>
    <span id="second" class="time" begin="5" dur="10">
      This text will never appear while the first line is active.
    </span>
  </t:priorityClass>
</t:excl>
...
```

The first line of text will appear immediately and display until the 15-second mark. At 5 seconds, however, the second line attempts to interrupt the show. lower="never" means that it cannot happen.

Attribute: pauseDisplay

- Version Compatibility: SMIL 2.0
- Possible Values: disable, hide, show

Suppose you have taken control of elements by use of the `<excl>` and `<priorityClass>` elements. You may have paused their presentation in the timeline, but what about their actual display? To fill this need, SMIL 2.0 offers the `pauseDisplay` attribute. When a child element of `<priorityClass>` is paused, a value of `disable` will cause the element to become disabled. This will vary depending on the SMIL implementation of the player or browser. But an example of a disabled element might be one that is "grayed out." `hide` will cause the element to disappear from view. `show`, the default value for `pauseDisplay`, will cause the element to show normally.

In RealPlayer, use of `pauseDisplay` with `peers` is shown next:

```
* * *
<excl>
  <priorityClass peers="pause" pauseDisplay="show">
    <video id="first" src="first.rm" />
    <video id="second" src="second.rm" />
  </priorityClass>
</excl>
* * *
```

If the second video clip becomes active, the first clip will pause. `pauseDisplay` reinforces the effect of `peers="pause"` by keeping the paused clip showing in the player.

Attribute: `peers`

- Version Compatibility: SMIL 2.0
- Possible Values: `stop`, `pause`, `defer`, `never`

If the `<priorityClass>` element is used as a child of a `<excl>` element, a set of attributes becomes available to help control the behavior of the grouped elements. This allows for presentation control as the user branches among the elements bundled together by `<excl>` and `<priorityClass>`. In essence, these attributes act like traffic lights directing the flow of the presentation.

Children elements of a `<priorityClass>` element are referred to as `peers`. The `peers` attribute controls how its siblings function when one of the children becomes active. `stop` is the default value for `peers`. If `stop` is specified, an active element is stopped as the newly active element begins. If `pause` is declared, an active element is paused as the newly active element begins. When the new element finishes or stops, the old active element resumes again. With `defer`, the

newly active element waits until the currently active element is finished. If `never` is specified, a newly active element does not begin if another element is already active.

In RealPlayer, `peers` is used in the following manner:

```
*. *.
<excl>
  <priorityClass peers="pause">
    <video id="first" src="first.rm" />
    <video id="second" src="second.rm" />
  </priorityClass>
</excl>
*. *.
```

Suppose the second video clip becomes active. Its peer, the first clip, will pause.

The markup for `peers` in Internet Explorer looks much the same:

```
*. *.
<t:excl>
  <t:priorityClass peers="pause">
    <span class="time" begin="0" dur="15">
      First line of text displayed.
    </span>
    <span class="time" begin="5" dur="10">
      Second line of text is displayed, interrupting the first.
    </span>
  </t:priorityClass>
</t:excl>
*. *.
```

The first line of text will appear immediately and display until the 5-second mark. At that point, the second line appears. As a peer of the second line of text, the first line pauses. In the case of this example, the first line of text disappears from the browser screen. After 10 seconds have elapsed, the second text line will disappear and the first will resume its count toward its 15-second duration.

Attribute: restart

- Version Compatibility: SMIL 2.0
- Possible Values: `always`, `whenNotActive`, `never`, `default`

If an element is to be restarted in the timeline, the `restart` attribute controls the behavior of that function. An `always` value means that the element can be restarted at any time. As it sounds, `whenNotActive` means the element cannot be

restarted until its active role in the timeline ends. `never` means the element can never be restarted. `default` means the element will grab its behavior from the `restartDefault` attribute defined for the block of grouped elements. The default value of the `restart` attribute, as you might expect, is `default`. If no `restartDefault` is specified, the result is that the `restart` attribute functions with a value of `always`.

Attribute: restartDefault

- Version Compatibility: SMIL 2.0
- Possible Values: `always`, `whenNotActive`, `never`, `inherit`

For a group of elements in a parent element, `restartDefault` sets the overall behavior for a restart in the timeline. Like the `restart` attribute, an `always` value means the associated element can be restarted at any time. `whenNotActive` allows the element to be restarted when it is not actively functioning in the timeline. And `never` means the element can never be restarted. `inherit` allows the `restartDefault` attribute to snag the value from the parent element's `restartDefault`. The default value of `restartDefault` is `inherit`.

Attribute: syncBehavior

- Version Compatibility: SMIL 2.0
- Possible Values: `canSlip`, `locked`, `independent`, `default`

If you have media objects playing at the same time, you also have the potential for synchronization issues. What happens if one media object loads smoothly and another stream does not? `syncBehavior` offers the author a set of behaviors that address this.

The `canSlip` value allows the timing of the element to slip in relation to other grouped elements. `locked` forces the element to keep in step with the other elements. With `independent`, the element runs on its own timeline. Even if other elements in the grouping stop or seek to a new location in the timeline, `independent` makes sure the associated element keeps on its own merry way. `default` matches `syncBehavior` to the value established for a group of elements by the `syncBehaviorDefault` attribute.

To use `syncBehavior` with `RealPlayer`, tuck the attribute in with the media object or time container:

```
<video src="myVideo.rm" syncBehavior="canSlip" />
```

It is important to note that the implementation of the SMIL player or browser can have a great deal to do with how these attributes are followed. A player using `canSlip` might just let the timeline slip or it might try to catch up by skipping ahead in the playback. You may need to do further research or experimentation to get a result that successfully matches your presentation.

Attribute: `syncBehaviorDefault`

- Version Compatibility: SMIL 2.0
- Possible Values: `canSlip`, `locked`, `independent`, `inherit`

If you have a lot of media objects playing at the same time, you have a need for a `syncBehavior` that is more global. `SyncBehaviorDefault` fulfills this mission. Attached to an element, `syncBehaviorDefault` establishes the default synchronization behavior for all children of that element. You can override this default behavior by specifying a `syncBehavior` in a given element. This is ideal for that one errant element that seems to need something different.

As with `syncBehavior`, the `canSlip` value will allow the element to slip in the timeline. `locked` forces the element to keep in step with the timeline. With `independent`, the element runs on its own timeline. Even if other elements in the grouping stop or seek to a new location in the timeline, `independent` makes sure the associated element keeps on its own merry way. `syncBehaviorDefault` swaps the default value for `inherit`. This value tells `syncBehaviorDefault` to inherit the `syncBehaviorDefault` value from a parent element.

To use `syncBehaviorDefault` with `RealPlayer`, place the attribute in with the time container that holds the media objects:

```
* * *
<par syncBehaviorDefault="locked">
  <video src="myVideo.rm" />
  <audio src="myAudio.rm" />
</par>
* * *
```

Attribute: `syncTolerance`

- Version Compatibility: SMIL 2.0
- Possible Values: An acceptable clock value or `default`

If you set your `syncBehavior` or `syncBehaviorDefault` with the `locked` value, you might want to hedge your bets by saying just how locked the locked value really is. For example, you might set `syncTolerance` to a clock value of `3s`, meaning 3 seconds. As long as the offending element stays timed within 3 seconds of the other elements, everything proceeds normally. If the errant element falls outside of that 3-second range, the player might drop data to catch the presentation up, or it might halt the presentation until the late element comes back up to speed. Like `syncBehavior`, a default value for `syncTolerance` reaches back to find out what the `syncToleranceDefault` value is.

`syncTolerance` in `RealPlayer` is applied to the time container that groups the elements:

```
* * *
<par syncTolerance="3s">
  <video src="myVideo.rm" syncBehavior="locked" />
  <audio src="myAudio.rm" syncBehavior="locked" />
</par>
* * *
```

Attribute: `syncToleranceDefault`

- Version Compatibility: SMIL 2.0
- Possible Values: An acceptable clock value or `inherit`

If you have a number of elements that you want to lock the synchronization to, `syncToleranceDefault` makes it easy to establish the tolerance across the entire group. Like `syncTolerance`, you can use an acceptable clock value to represent hours, minutes, seconds, or milliseconds, and like `syncBehaviorDefault`, `syncToleranceDefault` offers an `inherit` value to follow the guidance of `syncToleranceDefault` in a parent element.

Just like `syncTolerance`, `SyncToleranceDefault` in `RealPlayer` is used on the time container that groups the elements:

```
* * *
<par syncToleranceDefault="3s">
  <video src="myVideo.rm" syncBehavior="locked" />
  <audio src="myAudio.rm" syncBehavior="locked" />
</par>
* * *
```

Attribute: syncMaster

- Version Compatibility: SMIL 2.0
- Possible Values: true, false

If all you want to do is synchronize your elements to one master element, `syncMaster` is your choice. `syncMaster` is attached to the media element or time container to which you want the timeline locked. With a lead element identified, all other elements within that time container will follow its direction. But `syncMaster` is ignored if the element is paused.

Attribute: timeContainer

- Version Compatibility: SMIL 2.0 (Integration Set)
- Possible Values: par, seq, excl, none

With SMIL 2.0, Web multimedia authors now have the capability to incorporate SMIL into other XML languages. A great example of this is the HTML+TIME implementation of SMIL 2.0 in the Internet Explorer Web browser. With Internet Explorer, SMIL behavior appears right within the Web browser.

`timeContainer` attaches timing capability to XML elements. The attributes are the expected `par` (parallel), `seq` (sequential), and `excl` (exclusive) methods of grouping and timing elements that you would find in more traditional implementations of SMIL. `timeContainer` also offers a `none` value. This default value turns off the `timeContainer` function.

Obviously, you cannot use `timeContainer` as an attribute for an existing `<par>`, `<seq>`, or `<excl>` element. Beyond that restriction, it is up to the XML language author as to what elements `timeContainer` can be attached.

On first glance, `timeContainer` might seem a bit redundant. For example, you could use the following markup in Internet Explorer to display three banner ads in sequence:

```
...
<span class="time" timeContainer="seq">
  
  
  
</span>
...
```

Of course, you could also use a more traditional approach with normal use of a `<seq>` element. So why is `timeContainer` so handy? At times, you might want to leverage the time container aspects of SMIL against the power already built in to the host environment. For example, Internet Explorer supports a `<marquee>` element intended for scrolling text. With `timeContainer` attached, it is possible to scroll images across the screen while they change every few seconds:

```
* * *
<marquee class="time" timeContainer="seq" repeatCount="indefinite">
  
  
  
</marquee>
* * *
```

Now, an image will scroll across the Web page; every 5 seconds, the image will change to the next image in the sequence. The result is subject to your interpretation of taste. But it does quickly demonstrate the power available in a SMIL Integration Set.

Attribute: timeAction

- Version Compatibility: SMIL 2.0 (Integration Set)
- Possible Values: `intrinsic`, `display`, `visibility`, `style`, `class:classname`, `none`

The purpose of `timeAction`, as the name suggests, is to specify what action is taken on an element while it is active in the timeline.

`intrinsic` is the default value for `timeAction`. It basically tells the element that it should do its natural thing, whatever that may be, when the timing is applied.

When used with the `style` value, `timeAction` controls the application of a style as set by an inline `style` attribute. As shown in the following markup sample from Internet Explorer, you could set text to appear temporarily bold for a few seconds after the Web page loads:

```
* * *
<span class="time" style="Font-Weight:bold;" begin="0" dur="10"
  timeAction="style">
  <p>This text will be bold for the first ten seconds of display before
  reverting to normal.</p>
</span>
* * *
```

After the duration of 10 seconds has elapsed, the text returns to a normal nonbold style.

Similarly, `class` controls the application of a classname to the element during the associated timeline. The form for the `class` value is to specify the appropriate classname after `class` and a colon (as in `class:classname`). `none` turns off any timing effect on the element.

Finally, the `display` and `visibility` values of `timeAction` have very similar functions with slightly different flavors. Both `display` and `visibility` cause the element to appear while the timeline is active. Both `display` and `visibility` also cause the element to disappear when the timeline is finished. So what is the difference? With `display`, the element is not a part of the presentation until it is displayed. So what the user might see with `display` is a SMIL-based Web page that dynamically changes to make room for the element as it appears. When the element is no longer displayed, the Web page may dynamically change back to its old layout. With `visibility`, the element is a part of the presentation, but it is hidden until it is made visible. Therefore, the Web page layout would stay the same, but the element would appear in the position already set aside for it.

If you are used to Web pages sitting still on the screen, all this dynamic reformatting might seem a foreign concept. So let's look at a markup example that uses both values. Again, we'll use a sample from Internet Explorer. Suppose you had a two lines of text displayed on the screen:

```
...
<span class=time begin="0" dur="10" timeAction="visibility">
  <p>This first line of text will disappear after ten seconds.</p>
</span>
<p>This second line of text will stay in its original place on the
screen.</p>
...
```

Both lines of text will appear on the screen when the Web page loads. The first line of text will disappear from the screen after 10 seconds. But with `timeAction="visibility"` the second line will stay anchored to its original position on the screen.

Now let's try something a little different: -

```
...
<span class=time begin="0" dur="10" timeAction="display">
  <p>This first line of text will disappear after ten seconds.</p>
</span>
```

```
<p>This second line of text will shift up to occupy the space of the first  
→ line.</p>
```

Once again, both lines of text will appear on the screen when the Web page loads. And as with the previous example, the first line of text will disappear from the screen after 10 seconds. But now things change. With `timeAction="display"` the second line of text will jump up to fill the vacant hole left behind as the first line disappears. The Web page dynamically changes.

These two examples are pretty simple. You can imagine how drastic the changes would be if an entire paragraph of text or large graphic disappeared. With `timeAction="display"` the other page objects will reshuffle on the page as if the old elements never existed. In a strange way, `timeAction="display"` creates a kind of SMIL-based quicksand.

The Timing and Synchronization Module Summarized

Even in SMIL 1.0, Timing and Synchronization did the bulk of the work for the SMIL Web multimedia author. SMIL 2.0 greatly expands the earlier `<seq>` and `<par>` time containers with the highly interactive `<excl>` and a whole new set of attributes that allows for a great deal of fine-tuning of the presentation. Attributes take the power of SMIL time containers right into other XML languages, meaning you can now flavor your XHTML with SMIL. SMIL 2.0 authors will find that the language has now earned the Synchronized Multimedia portion of its title.



Bring on the Media: The SMIL Media Object Modules

From the previous chapters, you know that a SMIL document does not contain any media on its own. Instead, it references the source for a media element and activates the retrieval of the file contents. The SMIL document also informs the player as to how, where, and when to play back the media source file that has been retrieved. These source files, which can contain audio, video, text, streaming text, animations, still images, or even other SMIL files, are commonly referred to as *media objects*. Therefore, the Media Object modules provide the elements and attributes used to integrate these media files into a SMIL presentation.

SMIL 2.0 has slightly expanded the methods for handling media content that were available in SMIL 1.0, but in many ways, the new specification continues with what worked well previously. The BasicMedia module retains much of the functionality found in SMIL 1.0, and most of what is now reorganized into the other Media Object modules (such as MediaClipping, MediaClipMarkers, MediaAccessibility, and MediaDescription modules) also hails from the SMIL 1.0 specification. However, the MediaParam module and the BrushMedia module introduce two new elements: one that assists with attaching media-specific attributes to objects (*param*) and one that allows the painting of a single color in place of a media object (*brush*).

Although a few timing attributes are included with the Media Object modules, almost all timing and control in SMIL 2.0 is achieved through the use of the elements and attributes found in the Timing and Synchronization modules, which are covered in depth in Chapter 7, “Grouping Your Content: The Timing and Synchronization Module.” Although a presentation’s interactivity and hyperlinking is often carried out in association with media object elements and attributes, those functions are not discussed here; instead, they are discussed in Chapter 11, “Making It Interactive: The Linking Modules,” which focuses on SMIL 2.0’s Linking modules.

Conformance Profiles and the Media Object Modules

The naming and retrieving of media elements is essential to a SMIL presentation, and without the functionality of the BasicMedia module, there would be no media in the multimedia presentation. Therefore, it is not surprising to find that the SMIL 2.0 Language Profile and SMIL 2.0 Basic Language Profiles support all the Media Object modules in full.

Integration-set conformant language profiles, should they choose to make use of the MediaParam or BrushMedia modules, must support the elements of the BasicMedia module and define several of the terms describing attributes of the new brush and param elements as well. Otherwise, most of the attributes and values that have been supported since SMIL 1.0 will most likely continue to follow the syntax used to implement them using that earlier specification.

The Media Object Module Elements

The elements that make up the Media Object modules provide the means for introducing media files into a SMIL presentation. Without these essential elements and the attributes that describe them, there would be no visuals, soundtracks, narrations, HTML documents, or the capability to link to other related presentations. In addition, the new param element allows for a set of modifying attributes to be attached to a media object. These attributes are activated when the media file is called for in the presentation. We’ll go into each of the elements in the following section and then proceed to the attributes that work with them.

Element: <animation>, <audio>, , <ref>, <text>, <textstream>, <video>

- Version Compatibility: SMIL 1.0, SMIL 2.0
- Possible Attributes: clipBegin, clip-begin, clipEnd, clip-end, erase, mediaRepeat, sensitivity, src, type

These elements are the basis of how a SMIL document references and retrieves the media files that make up the presentation and, as such, create the backbone of the Media Objects modules. It is important to note that although these appear to be distinct types of media, this is not the manner by which the player determines which type of media is contained in the file. The player will instead determine the type of a media object from the type attribute or from the MIME-type information communicated by the server or the viewer's operating system.

In spite of this, the author of a SMIL presentation should attempt to use the element that most closely resembles the media object in question, and if in doubt, use the generic ref element to identify a media object. This increases the readability of the SMIL document and, in turn, increases the chance that the viewer will experience a media-rich presentation without delays, holes, or stoppages.

The animation element that is in this module is used to reference an animation such as a scalable vector graphics animation or an animated GIF. In contrast, the Animation modules discussed in Chapter 10, "Bringing Graphics to Life: The Animation Modules," allow for the control of specific attributes and elements of an animation in a more in-depth manner. Be sure to read through Chapter 10 to incorporate these capabilities and fully make use of SMIL 2.0's new functionality concerning animation.

As mentioned in this chapter's overview, anchors and links can be attached to visual media objects that are rendered onscreen. This is covered in depth in Chapter 11, which details the functionality of the SMIL 2.0 Linking modules.

Element: <brush>

- Version Compatibility: SMIL 2.0
- Possible Attributes: color, clipBegin, clipEnd, erase, mediaRepeat, sensitivity, type

The brush element, new to SMIL 2.0, gives the author the option of using a solid color in place of a media object. All the attributes of the regular media objects are supported, with the exception of the `src` attribute, which is ignored because there is no media object to be retrieved. However, a separate `color` attribute is added for this element because it is necessary for the implementation of this effect. In brief, the brush element paints a solid color on the screen and can be used to create a succession of color changes between media clips.

Element: <param>

- Version Compatibility: SMIL 2.0
- Parent Element: animation, audio, img, ref, text, textstream, video
- Possible Attributes: name, value, valuetype

Also new to SMIL 2.0 is the param element, which allows authors to attach a set of modifying attributes and values to a media object, which is then accessed as the media file is pulled into play. This is accomplished by enclosing one or more param elements inside a particular media object element (for instance, a video element). The param element attributes are then used to augment or modify the media object. The bare bones structure of this process follows, with the param element at hand to modify the media object with an `id` of poem. Note that the media object element is not closed until after the param element has been closed.

```
<ref id="poem" . . . >  
  <param . . . />  
</ref>
```

Some sample uses of the param element include attaching design parameters that are instance-specific to an object (such as adding a cloud to an animated sky for only that particular instance of the sky animation), providing a link to a resource that stores player-specific runtime values, or calling by name in media elements found elsewhere in the presentation.

The information provided by the `param` element and its attributes are intended only for a specific media object and its subsequent implementation in a player or browser. Therefore, it is outside the scope of the SMIL specification to define the syntax used to interpret the modifying information by that media file or its player. In fact, the names and values delivered by the `param` element are assumed to be understood by the media object in question, or by its player. No specifics are given as to how players should retrieve or resolve the data that is supplied by the `param` element.

However, the SMIL specification does define several key attributes for the `param` element and thus sets the basic methodology for these modifications to take place. We'll use the animated sky mentioned previously as an example to illustrate this process.

```
<ref src="http://www.whoknows.com/sky.blue">  
  <param name="weather" value="cloudy" valuetype="data"/>  
</ref>
```

The first attribute in `param`, `name`, calls up a modifying or augmenting parameter by name. This parameter, again, is assumed to be understood by the media object to which it is attached, or by its player; in other words, either the `sky.blue` animation itself or its designated player are expected to understand the parameter passed to it. In this example, suppose that our animated sky can be modified to display a variety of weather patterns. Our parameter for these possibilities could be called `weather` and have several values available to it, each of which would modify the animated sky in a particular fashion. The `value` attribute is used to specify which of a parameter's possible values to call forth. Let's say that two values are available: one called `cloudy` and one called `snow`. Because we want a cloud, not snow, our example specifies `cloudy` to be the value paired with `weather`. Last, the `valuetype` attribute is used to specify the type of data that will be supplied by the name-value pair. In our case, `cloudy` is evaluated and passed to the player as a string, which is then implemented. Any number of `param` elements can be attached to a media object, in any order, but they must come at the start of the media object element and be enclosed by it. More information on the attributes of the `param` element is provided in the attributes section of this chapter.

The Media Object Module Attributes

The Media Object modules attributes provide definition and information about the media objects included in a SMIL presentation. Some of the attributes are used to supply information for viewers who are visually or hearing impaired; other attributes supply information about the content itself, and others assist with the way the media objects are handled.

Attribute: `abstract`

- Version Compatibility: SMIL 1.0, SMIL 2.0 (deprecated)
- Parent Elements: `animation`, `audio`, `img`, `ref`, `text`, `textstream`, `video`
- Possible Values: author-generated text

The `abstract` attribute is described under the `MediaDescription` portion of the Media Objects module and is meant to describe, briefly, the content contained in the media element. In contrast to the information contained by the `alt` attribute, this information is not displayed as alternate content, but is used for media content descriptions when a table of contents is generated from a SMIL presentation.

Although supported, this attribute is deprecated in favor of the use of the informational attributes found in and supported by the Metainformation modules (see Chapter 5, "Data About Data: The Metainformation Module").

Attribute: `alt`

- Version Compatibility: SMIL 1.0, SMIL 2.0
- Parent Elements: `animation`, `audio`, `brush`, `img`, `ref`, `text`, `textstream`, `video`
- Possible Values: author-generated text

Defined within the `MediaAccessibility` module, the `alt` attribute specifies alternate text to be displayed when the viewer's player cannot display a particular media object or if a player has had its preferences set so as not to display the given media type. `alt` text can also be displayed along with the media object, although this is implementation dependent and is often used by assistive devices for viewers with disabilities. The `alt` description of the media object should be brief and meaningful so that it indicates what it is that the viewer is not seeing. It is strongly recommended that the author always include the `alt` attribute for a

given media element because viewer environments and support for the display of media types are widely variable.

In the example that follows, the alt value of photo of cat will be read to a viewer using an assistive device or be displayed as text if the image itself is not rendered because of viewer preferences or system limitations.

```

```

Attribute: author

- Version Compatibility: SMIL 1.0, SMIL 2.0
- Parent Elements: animation, audio, img, ref, text, textstream, video
- Possible Values: author-generated text

The author attribute identifies the name of the author of the media element itself. To identify the author of the entire presentation, use the author attribute found in the Metainformation modules.

In the example that follows, Jane Doe is identified as the author of the media clip with the id of wildthing; if she is also the author of the presentation, that information should be included in the head section using the metainformation elements and attributes.

```
<ref id="wildthing" author="jane doe" . . . />
```

Attribute: clipBegin, clip-begin, clipEnd, clip-end

- Version Compatibility: SMIL 1.0 (clip-begin and clip-end), SMIL 2.0
- Parent Elements: animation, audio, brush, img, ref, text, textstream, video
- Possible Values: npt, smpte, smpte-30-drop, smpte-25, marker

As mentioned in the introduction to this chapter, the MediaObject modules are primarily concerned with the basic handling of media objects, whereas the timing and synchronization of media elements is principally done by the elements and attributes found in the Timing and Synchronization modules (discussed in Chapter 7). The attributes defined in this section, which is formally identified as the MediaClipping module, are the exception, and provide SMIL authors with the tools to begin and end playback of a continuous media object from starting and

stopping points along its internal timeline. When used in conjunction with the Timing module, these attributes are applied first.

The smaller portions of the continuous media object that are created by these starts and stops are commonly referred to as *sub-clips*. `clipBegin` specifies the beginning of a subclip offset from the start of the media object, whereas `clipEnd` designates the point at which the subclip will stop playing, again measured from the start of the media object. `clip-begin` and `clip-end` offer the same functionality, but are deprecated in favor of the non-hyphenated attributes. In the following example, playback of the video object begins 10 seconds after the start of the clip and ends 25 seconds from the start of the clip.

```
<video src="movie.rm" clipBegin="10s" clipEnd="25s"/>
```

The values available for defining these attributes are designated as “a metric specifier followed by a related time value;” in the preceding example, the timing syntax is making use of the default time measurement of *npt*, or *normal playing time*, which represents SMIL clock values. The inclusion of *s* indicates seconds in this case. The metric specifiers available for defining time values are described in detail in the Timing and Synchronization modules (covered in Chapter 7) and include *smpte*, *smpte-30-drop*, *smpte-25*, *npt* (the default value) or *marker*. SMPTE timecode can be used for frame-level access and is measured in units that are specified as hours:minutes:seconds:frames.subframes, in which *sub-frames* are measured in one-hundredths of a frame. *smpte-30-drop* corresponds to the NTSC drop-frame timecode of 29.97 frames per second and *smpte-25* corresponds to the PAL standard of 25 frames per second. Again, for more information on time measurement, be sure to reference Chapter 7.

The use of the *marker* value is defined in the `MediaClipMarkers` module and allows for the use of named time points in a media object, if that media format supports the creation of marked time points. The value is expressed using the syntax `marker=#URI`, where the link is relative to the `src` attribute of the media object. In the following example, a video file in a format that supports marked time points has had a particular point marked as “night”. The `clipBegin` attribute expresses that the video object should begin at that named marker.

```
<video src="zoo.mov" clipBegin="marker=#night"/>
```

The SMIL 2.0 Language Profile includes support for all four attributes, whereas SMIL 1.0 recognizes only the hyphenated `clip-begin` and `clip-end`. On the

other hand, integration-set conformant language profiles that implement only the BasicMedia and MediaClipping modules recognize only the non-hyphenated `clipBegin` and `clipEnd`. Because SMIL 1.0 implementations will not recognize the non-hyphenated attributes, and profiles supporting only the SMIL 2.0 MediaClipping module may not recognize the hyphenated ones, you should include both styles of attributes when using any one of them. In addition, `clipBegin` and `clipEnd` take precedence over `clip-begin` and `clip-end`, respectively. In the following example, a SMIL 1.0 implementation will begin the video clip at 10 seconds after the start of the clip and ignore the `clipBegin` attribute entirely. A SMIL 2.0 Language Profile implementation and any language profile that supports the MediaClipping module will begin the video at 15 seconds and ignore the `clip-begin` attribute.

```
<video src="zoo.mov" clip-begin="10s" clipBegin="15s"/>
```

As an alternative to including both styles of attributes, the `switch` element outlined in the ContentControl modules can be employed. See Chapter 12, "Optimizing Content Delivery: The Content Control Modules," for more information regarding this functionality.

Attribute: color

- Version Compatibility: SMIL 2.0
- Parent Elements: `brush`
- Possible Values: hexadecimal colors or keyword colors

Defines the color that fills the screen when using the `brush` element.

Attribute: copyright

- Version Compatibility: SMIL 1.0, 2.0
- Parent Element: `animation`, `audio`, `img`, `ref`, `text`, `textstream`, `video`
- Possible Values: author-generated text

The `copyright` attribute contains the copyright notice for the content of the media element or object. To state the copyright notice for the entire SMIL presentation, use the `copyright` attribute found in the Metainformation modules.

Attribute: erase

- Version Compatibility: SMIL 2.0
- Parent Element: animation, audio, brush, img, ref, text, textstream, video
- Possible Values: whenDone, never

Used with any media element, the `erase` attribute controls whether a media object will stay onscreen after the effects of any timing have been completed. The default value of `whenDone` has the effect of removing the media after any applied timing is completed. Use of the value `never` will keep the media displayed in its last state until the region is reused or until the media is restarted. Any linking behavior will stay active as long as the links are displayed. In the following example, two media elements are displayed sequentially and kept onscreen until the presentation ends.

```
<seq>
  <video src="video1.mov" region="one" fill="freeze" erase="never"/>
  
</seq>
```

Attribute: longdesc

- Version Compatibility: SMIL 2.0
- Parent Element: animation, audio, brush, img, ref, text, textstream, video
- Possible Values: URI link

Another attribute found in the MediaAccessibility module, the `longdesc` attribute is a URI that links to a long text description describing the media object. The description should be supplementary to the information provided in the object's `alt` and `abstract` attributes, and if the object contains hyperlinks, it should also provide information about the linked content. In the following example, the `longdesc` attribute identifies the resource `wordy.html` as the document that describes the content of `fox` in detail.

```
<video id="fox" longdesc="wordy.html" alt="video of our fox" abstract="a
  short film about the daily life of a fox". . . />
```

Attribute: mediaRepeat

- Version Compatibility: SMIL 2.0
- Parent Element: animation, audio, brush, img, ref, text, textstream, video
- Possible Values: preserve, strip

Used with media types for which a distinct repeat value is found in their specification, such as animated GIFs or looped video sequences, the `mediaRepeat` attribute allows the SMIL author to override the repeat behavior intrinsic to the media type as the object is referenced. The default value of `preserve` leaves the repeat value of the media object intact. In this case, an animated GIF that has been intrinsically defined to repeat indefinitely would do so until it leaves the screen. A value of `strip` would remove the repeat behavior of a media object. In the example of the animated GIF used previously, the GIF would display only once in spite of its intrinsic indefinite repeat value. The following code indicates that the animated GIF `repeatme.gif` will be stripped of its indefinite loop and play only once.

```
<animation src="repeatme.gif" mediaRepeat="strip" . . . />
```

To use this attribute, a media type must have a formal specification of the exact repeat value to which the `mediaRepeat` attribute will apply, or the media type viewer must provide an interface for controlling the repeat value of the media. When using `mediaRepeat` with the SMIL Timing module attributes of `repeatCount` and `repeatDur`, `mediaRepeat` will be applied first with the Timing attributes controlling the repeat behaviors subsequently.

Attribute: name

- Version Compatibility: SMIL 2.0
- Parent Element: param
- Possible Values: implementation-specific (or player-specific) values

The `name` attribute declares the name of a runtime parameter for a media object and is expected to be understood by the object's player. The `name` attribute is paired with the other `param` attributes of `value` and `valueType`. The following example calls up the parameter `weather`, pairs it with a value of `cloudy`, and identifies that the information will be passed to the player as data.

```
<param name="weather" value="cloudy" valueType="data" />
```

Attribute: readIndex

- Version Compatibility: SMIL 2.0
- Parent Elements: animation, audio, brush, img, ref, text, textstream, video
- Possible Values: positive integers

The `readIndex` attribute presents the order in which assistive devices read aloud the content of any `longdesc`, `alt`, and `title` attributes of a given media element in relation to other media elements that are being presented simultaneously. The media object with the lowest `readIndex` value will have its description(s) read first, followed sequentially by objects with higher `readIndex` values. The default value of zero results in an element's descriptions being read last. The numbers assigned for `readIndex` values need not be sequential and can be randomly assigned. If two elements have identical `readIndex` values, their descriptions will be read in the order that they appear in the SMIL document.

In the following example, assistive devices will read aloud any accompanying `alt`, `longdesc`, or `title` values of the media object `cow` first because its `readIndex` value is the lowest. The values of those attributes for the audio object `moo` will be read next. In the case of `field` and `captions`, which have identical `readIndex` values, any descriptive values would be read in the order in which they appear in the document; in this case, `field` would be dealt with first, followed by `captions`.

```
<par>
  <ref id="cow" readIndex="2" . . . />
  <audio id="moo" readIndex="3" . . . />
  <video id="field" readIndex="7" . . . />
  <text id="captions" readIndex="7" . . . />
</par>
```

Attribute: src

- Version Compatibility: SMIL 1.0, SMIL 2.0
- Parent Element: animation, audio, img, ref, text, textstream, video
- Possible Values: URI link

The `src` attribute provides the address of the media object that is being called up by the SMIL document. The URI provided, whether relative or absolute, is used

to locate and retrieve the media clip and present it for playback. This attribute is not required, but no media will be retrieved or presented for a media element that does not have a `src` value; however, the object will be given an intrinsic duration of zero and calculated into any timing that occurs in the presentation. In the example below, the media object `moon` is retrieved from a source external to the presentation using absolute addressing, whereas `sun` is retrieved using relative addressing from a file that exists locally to the presentation in a directory named `images`.

```
<par>
  
</par>
```

Attribute: sensitivity

- Version Compatibility: SMIL 2.0
- Parent Element: `animation`, `audio`, `brush`, `img`, `ref`, `text`, `textstream`, `video`
- Possible Values: `opaque`, `transparent`, percentage value

The `sensitivity` attribute allows the SMIL author control over whether a particular media object is sensitive to user-interface selection events, such as hyperlink activation, or whether the user event is passed onto media objects that are lower in the display hierarchy. The default value of `opaque` indicates that the entire area of the media object is sensitive to user interactivity. A value of `transparent` passes an event down to media on layers that are beneath the top-layer media.

In the following example, the region `onTop` has a `z-index` value higher than the `z-index` of the region `behind`, which results in `behind` being displayed behind the region `onTop`. If `picture`, which is a media object displayed in `onTop`, was given (or left with) the default `sensitivity` value of `opaque`, it would be sensitive to user interaction. However, because it has been assigned a value of `transparent`, any links or clickable regions in `links` (displayed in `behind`) will be available for the viewer to activate.

```
<layout . . . />
  <region id="onTop" z-index="7" . . . />
  <region id="behind" z-index="1" . . . />
</layout>
. . .
```

```
<body>
  
  <ref src="links" region="behind" . . . />
</body>
```

If the rendered media supports an alpha channel, and the opacity of the media is less than the value declared using the `sensitivity` attribute, the media element is considered as if it were declared transparent and passes the event down to media lower in the display hierarchy. In other words, if the media object is only 30% opaque, and the `sensitivity` value is set lower than 30%, the object is considered transparent. If the alpha channel sets the opacity of the object at more than the percentage declared using the `sensitivity` attribute, the media object is considered opaque and is available for user interaction.

Attribute: `title`

- Version Compatibility: SMIL 1.0, SMIL 2.0
- Parent Elements: `animation`, `audio`, `brush`, `img`, `ref`, `text`, `textstream`, `video`
- Possible Values: author-defined text

The `title` attribute is intended to inform about the element to which it is attached and should include information that is brief, yet meaningful. Implementation depends on the player or browser; for example, the information may be displayed in a title bar or viewed as a mouse is rolled over an element.

How this information is displayed varies not only on the player, but also on the browser version and the operating system of the user. For example, on a Mac using RealPlayer, after a media object is activated, the `title` is displayed in a rolling title bar. In Internet Explorer on a Mac, the title is displayed in the top of the window.

Attribute: `type`

- Version Compatibility: SMIL 1.0, SMIL 2.0
- Parent Element: `animation`, `audio`, `brush`, `img`, `ref`, `text`, `textstream`, `video`
- Possible Values: depends on the protocol of the `src` attribute for the media element (see the following)

The `type` attribute, although not required, identifies the content type of the media object referenced by the `src` attribute. Very similar to MIME types, this attribute can directly influence the type of helper application that is used to play back media. Although SMIL allows authors to use a variety of elements, such as `animation`, `audio`, or `video`, to introduce media to the presentation, the element an author uses to enclose the media does not really declare the kind of media the media object contains. The `type` attribute specifies the kind of media that the media object contains, as well as what helper application may be used to play back the file.

The way this is done depends on the protocol used to deliver the file or the `src` value of the media object. In the following example, the media object `newsvideo` is delivered using RTSP (for more information on transfer protocols, see Chapter 2, "SMIL Authoring"). Because it uses this protocol for delivery, the `type` attribute is used to declare that the content be selected and, if the type is unknown to the current playback engine, that it be played back using the playback engine declared by the `type` attribute. In this case, `application/x-pn-realmedia` declares that if the viewer's media player doesn't know what to do with a file with the extension `.rm`, the playback engine to use is RealPlayer.

```
<par>
  <video id="newsvideo" src="rtsp://realserver.wherever.com/video.rm"
    type="application/x-pn-realmedia" . . . />
  <audio id="headlines" src="http://www.mydomain.com/captions.mov"
    type="audio/quicktime" . . . />
</par>
```

The media object `headlines` is delivered using HTTP as its delivery protocol; in this case, the `type` attribute influences the selection of the helper application to play back the file. Although the viewer's browser may have its preferences set to launch another audio tool to play back `.mov` files, the `type` attribute will force the file to play back using QuickTime.

In the case where the content is available in multiple formats, implementations may use the `type` attribute to influence which format is used.

Attribute: value

- Version Compatibility: SMIL 2.0
- Parent Element: `param`

- Possible Values: author-defined text, a URI link or the `id` attribute of another media element

The `param` element sets up parameters for a given media object that are pulled into play when a media object is called by a presentation. The `param`, `name`, `value`, and `valuetype` attributes are used to modify or augment the media object. The parameter is called by name by the media object, and the `value` attribute is used to select from a list of possible values to pair with the named parameter. The values, in turn, are used to modify the media object. In the following example, the `param` attributes describe a runtime parameter with the name of `weather` that is used to modify the media object `sky.blue`, whose paired value of `cloudy` dictates that a cloud be included in the `sky.blue` animation.

```
<ref src="http://www.whoknows.com/sky.blue">  
  <param name="weather" value="cloudy" valuetype="data"/>  
</ref>
```

Attribute: valuetype

- Version Compatibility: SMIL 2.0
- Parent Element: `param`
- Possible Values: `data`, `object`, `ref`

The attribute `valuetype` defines the type of information that will be passed on to a media element's player when a named parameter is called up using the `param` element. The default value for the `valuetype` attribute is `data`, which means that the information specified by the `value` attribute will be evaluated and passed to the media object's player as a string. The preceding example (under the attribute `value`) is an example of such a `valuetype`.

If the `value` attribute of the `param` element is the `id` attribute of another media object within the SMIL document, `valuetype` is assigned the value of `object`. In the following example, a media object has previously been assigned the `id` of `musik` and is called up in the runtime parameter named `sound`.

```
<param name="sound" value="musik" valuetype="object"/>
```

When the `valuetype` attribute is given a `ref` value, this specifies a URI or link to a resource where runtime values are stored. This URI must be passed unresolved and also necessitates the use of an additional attribute called `type`, which

specifies the type of content found in the named resource. In the example that follows, a set of runtime parameters are stored at `http://www.surlygrrl.com/params.txt`. The parameter with the name of `betsy` and the `value` type of `ref` links the media object specified in the `video` element to these parameters. The `type` attribute is used to identify the resource as having the content type (or MIME type) of `text/plain`.

```
<video . . .>
  <param name="betsy" value="http://www.surlygrrl.com/params.txt"
  value="ref" type="text/plain"/>
</video>
```

Attribute: `xml:lang`

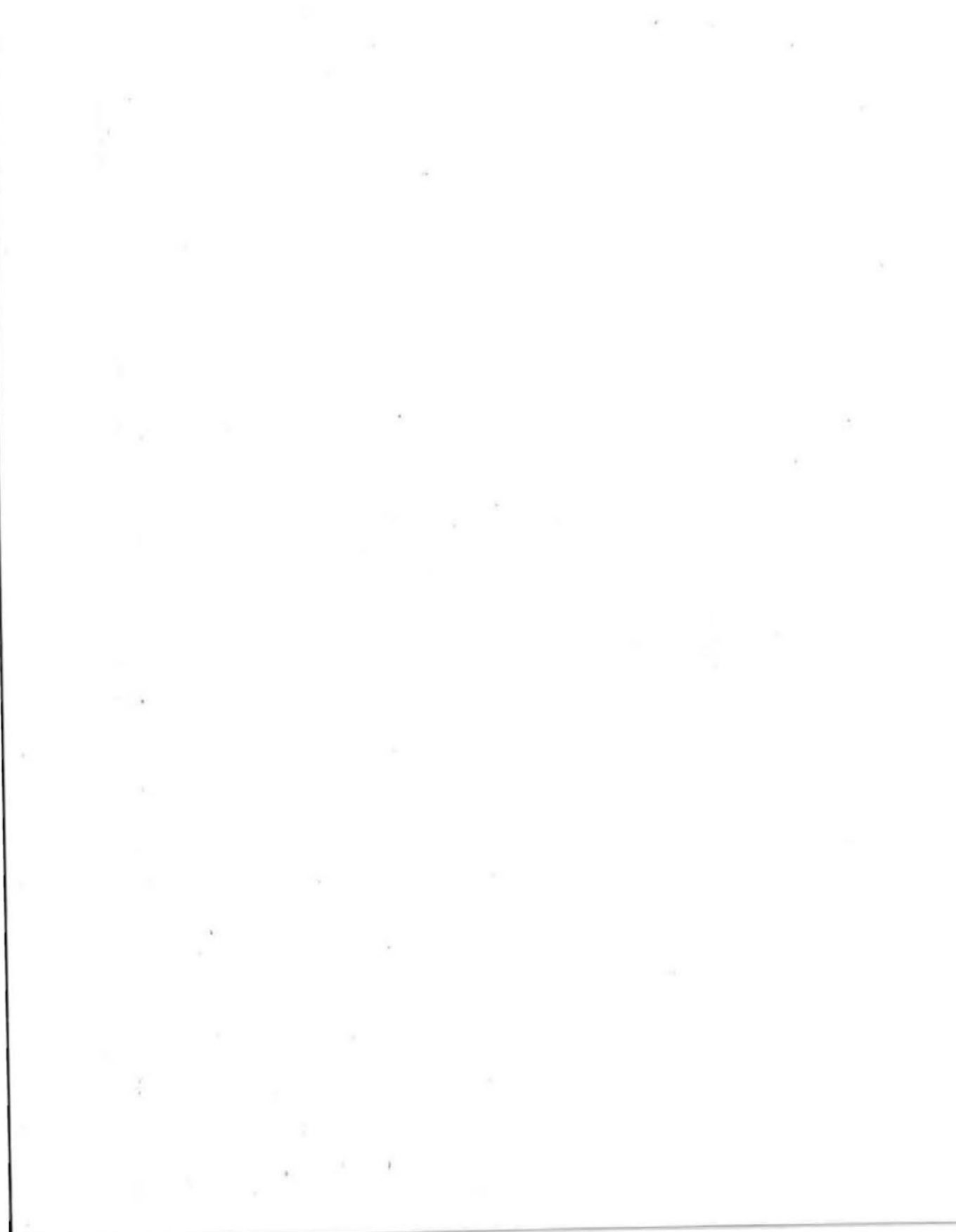
- Version Compatibility: SMIL 2.0
- Parent Element: `animation`, `audio`, `img`, `ref`, `text`, `textstream`, `video`
- Possible Values: valid language codes (with options from Ami to English to Klingon to Xiang)

Included in the `MediaDescription` module and as covered in the XML specification, this attribute identifies the natural or formal language of the media element. The information provided by this attribute concerns the media object's language and is independent of what SMIL players may do with the information. It is not to be confused with the `test` attribute of `systemLanguage`, which tests for the viewer's language preference. In the following code snippet, the language spoken in the audio clip `speech.aiff` is identified as German (`de` being the language code for `deutsch`, or German).

```
<audio src="speech.aiff" xml:lang="de" . . . />
```

The Media Object Modules Summarized

The best multimedia presentations tend to offer media-rich experiences for the viewer by combining a variety of media elements with interactivity, often over time. To effectively create such a presentation, media objects must first be referenced, retrieved, and readied for playback. The Media Object modules in SMIL 2.0 and the media resourcing functionality found in the SMIL 1.0 specification provide the tools to do just that. When the Media Object modules are combined with others, such as the Timing and Synchronization modules or the Linking modules, the SMIL author is well on the way to creating a media-rich experience for the viewer.



Part IV



Appendices

- A. SMIL 2.0 Modules, Elements, and Attributes 345
- B. Online Resources 357
- Glossary 363

Part IV

100 100 100 100 100 100 100 100 100 100

100 100 100 100 100 100 100 100 100 100

Appendix

100 100 100 100 100 100 100 100 100 100

100 100 100 100 100 100 100 100 100 100



Appendix A

SMIL 2.0 Modules, Elements, and Attributes

The following list is a chapter-by-chapter breakdown of the SMIL 2.0 modules, an alphabetical listing of their corresponding elements, and their associated attributes.

Structure Module (See Chapter 4)

Elements:

body	Contains content and pathways to content, as well as information related to the timing of the presentation.
head	Typically the home of such areas as layout and metainformation, this element contains information about the presentation that is not related to the timing or display of the media elements.
smil	The outer container element of a host-language conformant SMIL document.

Attributes:

class	Similar to id, class has the added flexibility of allowing the identifier to be used across several objects and elements.
id	Attaches a unique identifier to any element or object so that another element or object can reference the original element or object for another purpose (such as to enable animation, interactivity, and so on).

xml:lang	Specifies the language code that determines spoken or written language presented in the content.
xmlns	Declares the namespace of the SMIL document, often used to enable additional player-specific features.
title	Used to let the user know something about the object or element to which it is attached.

Metainformation Module (See Chapter 5)

Elements:

meta	Used to provide information about the content in the presentation.
metadata	A highly extensible approach to providing very detailed information about the content in the presentation.

Attributes:

content	Offers descriptive content that matches the type of metainformation identified with name.
name	Used with content, this attribute specifies the name for the type of metainformation supplied.

Layout Modules (See Chapter 6)

Elements:

layout	Used to create a graphical layout and control audio playback.
region	Designates space for media playback or layout.
regPoint	A registration point for arranging regions and media.
root-layout	Defines the overall screen dimensions for a presentation.
topLayout	The top-level layout when using multiple display windows or layouts.

Attributes:

background-color	Deprecated in favor of backgroundColor.
backgroundColor	Used to designate a background color for a presentation display area.
bottom	Used to declare the bottom edge or point of a display area.
close	Provides control for closing multiple display windows.
fit	Designates how media is handled when it does not fit into its assigned display area.
height	Used to declare the height of a display area.
left	Designates the left edge of a display area.

<code>open</code>	Provides control for opening multiple display windows.
<code>regAlign</code>	Used with media objects to precisely align the element with a registration point.
<code>regPoint</code>	Refers to previously defined <code>regPoint</code> by <code>id</code> value.
<code>region</code>	Refers to previously defined region by <code>id</code> value or <code>regionName</code> .
<code>regionName</code>	A unique identifier for a region.
<code>right</code>	Designates the right edge or point of a display area.
<code>showBackground</code>	Determines whether a background color will be visible.
<code>soundLevel</code>	Used to control the volume of audible media.
<code>top</code>	Designates the top edge or point of a display area.
<code>type</code>	Identifies the layout language to be used by the presentation.
<code>width</code>	Declares the width of a presentation space.
<code>z-index</code>	Used to identify the layer on which the display space or media object occurs.

Timing and Synchronization Module (See Chapter 7)

Elements:

<code>excl</code>	Declares an exclusive time container where one element or time container from a group of elements or time containers is played exclusively.
<code>par</code>	Declares a parallel time container where two or more elements or time containers can run at the same time.
<code>seq</code>	Declares a sequential time container where one element or time container follows another in sequence.
<code>priorityClass</code>	Used with an <code>excl</code> element to control the playback characteristics of other elements and time containers when an element or time container is chosen to play exclusively.

Attributes:

<code>begin</code>	Specifies the begin time or event for an element or time container.
<code>dur</code>	Specifies the duration time for an element or time container.
<code>end</code>	Specifies the end time or event for an element or time container.
<code>endsync</code>	Synchronizes the end of children in a <code><excl></code> or <code><par></code> group based on the behavior of one of the elements or objects in the group.
<code>fill</code>	Describes how an element or object looks on its end in a timeline.

fillDefault	Describes the default fill behavior for a block of elements.
higher	Used with <priorityClass> to describe the playback characteristics of parent elements or objects when another element or object is playing exclusively.
lower	Used with <priorityClass> to describe the playback characteristics of child elements or objects when another element or object is playing exclusively.
max	Defines the maximum active duration for an element.
min	Defines the minimum active duration for an element.
pauseDisplay	Used with <excl> and <priorityClass> to control the display of elements or object paused when an element or object is playing exclusively.
peers	Used with <priorityClass> to describe the playback characteristics of peers when another element or object is playing exclusively.
repeat	Allows an element to repeat for a specified whole number of times.
repeatCount	Allows an element to repeat for a specified whole or fractional number of times.
repeatDur	Declares that an element will continue to repeat until a specified duration ends.
restart	Controls the behavior of how an element can be restarted in a timeline.
restartDefault	Controls the default behavior of how a block of elements can be restarted in a timeline.
syncBehavior	Controls how elements and objects stay synchronized under varying network conditions.
syncBehaviorDefault	Declares a default for a block of elements that controls how elements and objects stay synchronized under varying network conditions.
syncMaster	Synchronizes the playback of all elements to one master element.
syncTolerance	Specifies how tightly or loosely elements and objects stay synchronized under varying network conditions.
syncToleranceDefault	Declares a default for a block of elements that specifies how tightly or loosely elements and objects stay synchronized under varying network conditions.
timeAction	Specifies the action that is taken on an element when it is in the timeline.
timeContainer	Allows timing and synchronization elements to be used in another XML-compliant language (such as XHTML).

Media Object Modules (See Chapter 8)

Elements:

<code>animation</code>	Used to call a media source file—usually animated vector graphics or other animated format—into a presentation.
<code>audio</code>	Used to call a media source file—usually with audio content—into a presentation.
<code>brush</code>	Allows a solid color to be used as a media object.
<code>img</code>	Used to call a media source file into a presentation, usually containing a still image.
<code>param</code>	Declares a runtime parameter for a media object. Often used to pass situation-specific adjustments to a media object.
<code>ref</code>	A generic catch-all element used to call a media source file into a presentation.
<code>text</code>	Used to call a media source file consisting of text or HTML into a presentation.
<code>textstream</code>	Used to call a media source file that contains moving text into a presentation.
<code>video</code>	Used to call a media source file—usually with video content—into a presentation.

Attributes:

<code>abstract</code>	A brief description of a media object that can be used when generating a table of contents.
<code>alt</code>	Specifies text to be displayed in place of a media object. Also used by assistive devices.
<code>author</code>	Identifies the author of the media object.
<code>clip-begin</code>	Deprecated in favor of <code>clipBegin</code> .
<code>clipBegin</code>	Specifies the start of playback offset from the start of the media object.
<code>clip-end</code>	Deprecated in favor of <code>clipEnd</code> .
<code>clipEnd</code>	Specifies the stop of playback offset from the start of the media object.
<code>color</code>	Defines the color that is painted when using the brush element.
<code>copyright</code>	Declares the copyright notice for the media object.
<code>erase</code>	Determines whether a media object will continue to display after it has reached the end of its duration.
<code>longdesc</code>	A URI link that leads to a long text description of the media object.
<code>mediaRepeat</code>	Used with media objects that loop either to support the looped behavior of the object or to override the looping behavior.

name	Names a runtime parameter declared by the <param> element.
readIndex	Dictates the order in which assistive devices will read aloud the alt, title, and longdesc values, if declared.
src	Provides the address used to locate and retrieve a media object.
sensitivity	Declares whether user actions are applied to the media object that is closest to the surface in a layered environment.
title	Provides a very brief description about the media object.
type	Identifies the MIME type of a media object and can be used to influence which helper application is used to play back a media file.
value	Used in conjunction with the name and valuetype attributes to define a runtime parameter for a media object.
valuetype	Defines the type of information being passed to a player using the param element along with the name and value attributes.

Transition Effects Module (See Chapter 9)

Elements:

transition	Used to define a transition class that is applied to media objects in the body of the presentation.
transitionFilter	Animates the progress of a transition and allows for full control over the timing of the progress of a transition.

Attributes:

begin	Specifies the begin time for a <transition> or <transitionFilter>. Defined fully in the Timing and Synchronization modules (Chapter 7).
borderColor	Specifies the content of a border generated along a wipe edge.
borderWidth	Specifies the width of a border generated along a wipe edge.
by	Specifies a relative offset value for the progress of a transitionFilter. Defined in the BasicAnimation module (Chapter 10).
calcMode	Declares the interpolation mode of the progress of the <transitionFilter>. Defined fully in the BasicAnimation module (Chapter 10).
direction	Specifies the direction that a transition will run.
dur	Declares the duration of a <transition> or <transitionFilter>.
end	Specifies the end time for a transition or <transitionFilter>. Defined in the Timing and Synchronization module (Chapter 7).
endProgress	Declares the amount of progression through a <transition> at which to end execution.

<code>fadeColor</code>	Specifies the start or end color when using the type attribute value of "fade" with a subtype value of "fadeToColor" or "fadeFromColor".
<code>from</code>	Declares the amount of progression through a <code><transitionFilter></code> at which to begin execution.
<code>href</code>	Indicates the target element to which a <code><transitionFilter></code> is applied using a URI.
<code>horzRepeat</code>	Declares how many times to perform the given transition along a horizontal axis.
<code>mode</code>	Indicates whether the parent element will transition in or out.
<code>repeatCount</code>	Allows an element to repeat for a specified whole or fractional number of times. Fully defined in the Timing and Synchronization module (Chapter 7).
<code>repeatDur</code>	Declares that an element will continue to repeat until a specified duration ends. Fully defined in the Timing and Synchronization module (Chapter 7).
<code>startProgress</code>	Declares the amount of progression through a <code><transition></code> at which to begin execution.
<code>subtype</code>	Defines the subtype of a transition, determined by the type attribute of the transition in question.
<code>targetElement</code>	Specifies the element to which a <code><transitionFilter></code> is applied using the identifier (or <code>id</code> attribute value) of the element.
<code>to</code>	Specifies the amount of progression through a <code><transitionFilter></code> at which to end execution.
<code>transIn</code>	Indicates that a transition should begin at the beginning of the media object to which it is attached.
<code>transOut</code>	Indicates that a transition should end at the end of the media object to which it is attached, or at the end of its fill state.
<code>type</code>	Declares the type or family of a transition. Required.
<code>values</code>	A semicolon-separated list of one or more values specifying the progress of the <code><transitionFilter></code> .
<code>vertRepeat</code>	Defines how many times to perform a given transition along a vertical axis.

Animation Modules (See Chapter 10)

Elements:

<code>animate</code>	A generic animation element that starts an animated effect.
<code>animateColor</code>	Used to animate the color change of an element.
<code>animateMotion</code>	Used to animate the motion of an element.
<code>set</code>	Changes an attribute to a set value for a limited period of time.

Attributes:

accumulate	Controls whether the animation is cumulative.
actuate	Used with the href attribute; follows the link to the target element automatically.
additive	Controls whether the animation is additive.
attributeName	Specifies the name of the target attribute.
attributeType	Specifies the type of the named attribute.
by	Specifies the relative offset value of the animation.
calcMode	Determines the interpolation approach for the animation.
from	Declares a starting point for an animation.
href	Declares the target element to be animated by using an XLink locator.
keySplines	Defines control points that describe the speed (easing) of a spline animation.
keyTimes	Controls the pacing of a spline animation.
origin	Used by <animateMotion> to declare the origin of motion for the animation.
path	Uses coordinate values to describe the curve used by spline animation in <animateMotion>.
show	Used with the href attribute; specifies that the link does not include additional content.
targetElement	Used to specify which element the animation effect is being targeted to.
to	Declares the ending value for the animation.
type	Used with the href attribute; declares the type of XLink used.
values	Used to provide values needed by a supporting element or attribute.

Linking Modules (See Chapter 11)

Elements:

a	Allows for a hyperlink to be associated with a media object that has a visual display.
anchor	Deprecated in favor of the area element.
area	Used to define a portion of a media object's visual display area as a hyperlink.

Attributes:

<code>accesskey</code>	Defines a keyboard key that can be used to activate a hyperlink.
<code>actuate</code>	Declares whether user action is necessary for a hyperlink to be activated.
<code>alt</code>	Provides alternative text to be displayed along with a hyperlink or in place of it. Also used by assistive devices for the visually impaired.
<code>coords</code>	Used to identify a set of spatial coordinates that define a clickable space on the display area of a media object.
<code>destinationLevel</code>	Sets the audio level of a resource when a link is followed.
<code>destinationPlaystate</code>	Determines whether a resource begins playing when a link is followed.
<code>external</code>	Identifies whether the destination resource should be opened using the current application or an external one.
<code>fragment</code>	Allows for a SMIL presentation to overlay and/or override links in embedded media.
<code>href</code>	Identifies the URI of a destination resource in a hyperlink.
<code>nohref</code>	Specifies that a space declared using the area element does not have a URI associated with it.
<code>shape</code>	Designates the clickable shape of an area on the display surface of a media object.
<code>show</code>	Controls how the source presentation display responds when a hyperlink is followed.
<code>sourceLevel</code>	Sets the audio level of the presentation that contains a hyperlink when the link is followed.
<code>sourcePlaystate</code>	Determines the temporal behavior of the source presentation when a hyperlink is followed.
<code>tabindex</code>	Allows for the manual assignment of a tab order for links within a presentation should they be navigated by keyboard.
<code>target</code>	Identifies the display environment into which a destination resource should be opened.

Content Control Modules (See Chapter 12)**Elements:**

<code>customAttributes</code>	Allows for the definition of one or more author-defined custom test attributes.
<code>customTest</code>	Defines specific custom test attributes.

prefetch	Allows for media files to be retrieved, or preloaded, prior to being called up by the presentation.
switch	Used to enclose a set of presentation alternatives from which one can be chosen for implementation, based on the evaluation of its accompanying test attributes.

Attributes:

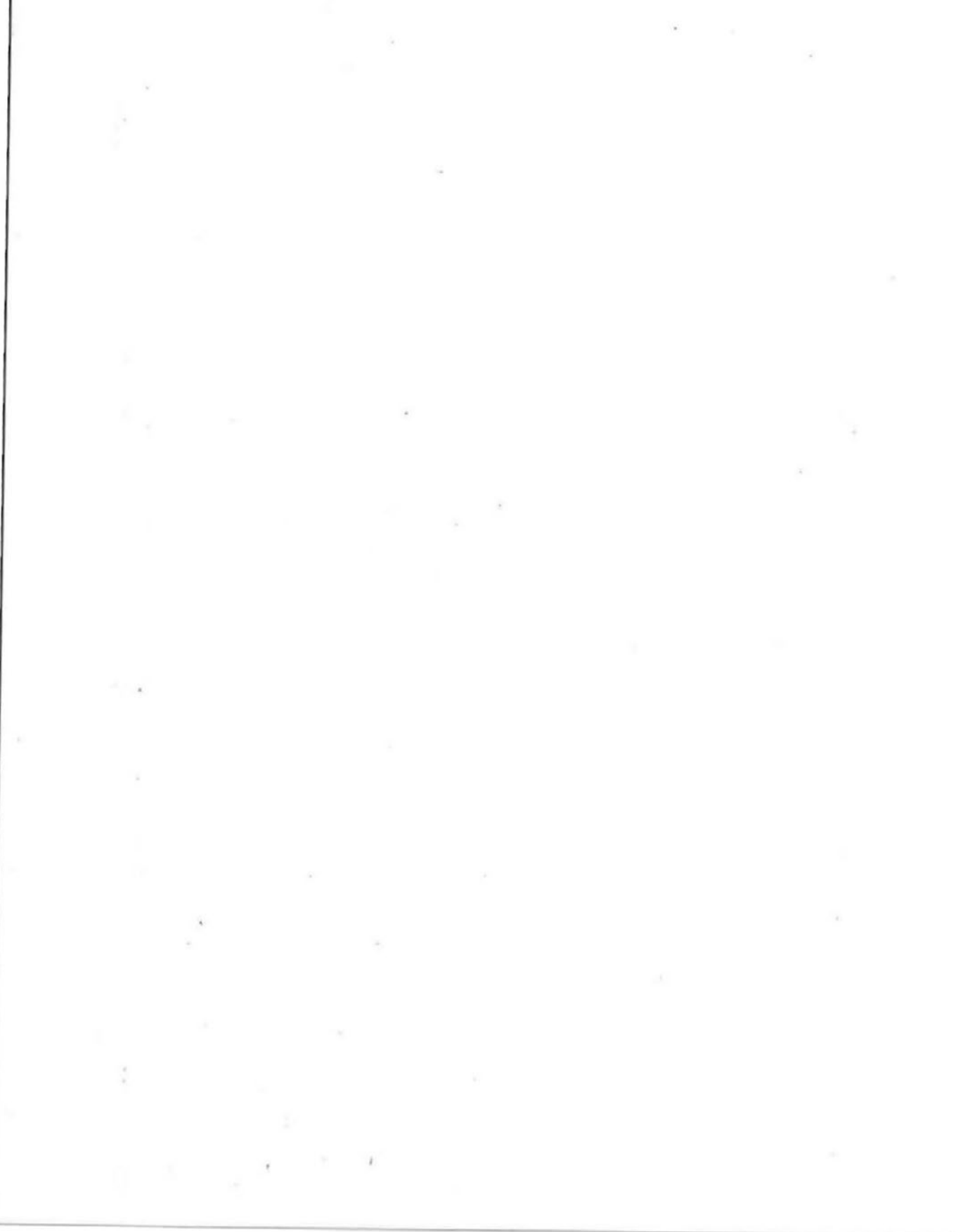
bandwidth	Defines how much bandwidth should be used to prefetch files.
customTest	Used to call a custom test attribute by its identifier.
defaultState	Determines the initial state for a named custom test variable.
mediaSize	Declares how much of a file to prefetch based on the size of the file.
mediaTime	Declares how much of a file to prefetch based on the duration of the file.
override	Used to determine whether a user has the capability to override the initial state of a custom test variable.
skip-content	Determines whether the content of an element is skipped or evaluated.
system-overdub-or-caption	Deprecated in favor of systemOverdubOrSubtitle and systemCaptions.
systemAudioDesc	A test attribute used to determine whether closed audio descriptions should be rendered.
systemBitrate	A test attribute used to deliver presentations that match the user's bandwidth capabilities.
systemCPU	A test attribute that can assist with specifying the CPU on which the SMIL player is operating.
systemCaptions	A test attribute that allows for the display of text equivalent to the audio portion of a presentation.
systemComponent	A test attribute that provides a URI that is used to identify a component of the playback system.
systemLanguage	A test attribute that can be used to match the presentation's language to that of the viewer.
systemOperatingSystem	A test attribute that can assist with specifying the operating system of a viewer.
systemOverdubOrSubtitle	A test attribute that determines whether a viewer prefers overdubbing or captioning, when available.
systemRequired	A test attribute used to provide an extension mechanism for new elements or attributes.

<code>systemScreenDepth</code>	A test attribute that can be used to specify the depth of the screen color palette required for display.
<code>SystemScreenSize</code>	A test attribute that can be used to specify the screen size necessary for the given element to be displayed.
<code>uid</code>	Identifies a URI identifier for a given custom test that is to be used persistently.

Time Manipulations Module (See Chapter 13)

Attributes:

<code>accelerate</code>	Accelerates the timeline of a media object or time container from stopped to full speed.
<code>autoReverse</code>	Reverses the playback of a timeline of a media object or time container when its end is reached.
<code>decelerate</code>	Decelerates the timeline of a media object or time container from full speed to stopped.
<code>speed</code>	Slows down or speeds up the timeline of a media object or time container.



Appendix B

Online Resources

The Definitive Authority: The World Wide Web Consortium (W3C) SMIL Pages

<http://www.w3.org/AudioVideo/>

Maintained by the standards body for the Web and the authors of SMIL, this site is the official resource for SMIL 1.0 and 2.0. It includes links to the specification documentation for both versions, as well as the history of SMIL and other useful resources.

Companion Site for This Book

Technology moves forward quickly. As we write this book, the SMIL community is constantly gaining new browsers, players, tools, and authoring techniques. But at some point with each chapter, we have to close the file and send what we have to the publisher. SMILBook is our way of keeping you up to date on the world of SMIL.

SMILBook

<http://www.smilbook.com/>

Check here to keep posted on the changes and developments that occurred after this book was published.

Downloadable Media Players

Media players are necessary for the playback of SMIL presentations and their associated media files. The players that support playback of SMIL presentations are available using the following URLs.

Supported platforms: Mac and Win-98/Me/NT4/2000

RealPlayer 8.0

<http://www.real.com/>

This version of RealPlayer supports playback of SMIL 1.0 presentations, as does the earlier G2 version. The next generation of this player, called RealONE, was released as a preview in late September of 2001. When the final version of RealONE is released, it will be available at this same location.

QuickTime 5.0

<http://www.apple.com/quicktime/download/>

Apple's QuickTime player can be used to playback SMIL 1.0 presentations. The next version of this player, due out in early winter of 2002, is expected to support SMIL 2.0.

Internet Explorer 6

<http://www.microsoft.com/windows/ie/default.htm>

Microsoft's browser is able to play back SMIL 2.0 presentations that are authored using its HTML+TIME language profile.

GRiNS-SMIL 2 Player (Oratrix)

<http://www2.oratrix.nl/Products/G2P?zone=G2P>

Oratrix was one of the first companies to create a player to support SMIL 2.0 files. Available as a free download for both Macintosh and Windows platforms, the GRiNS player is part of a commercial venture by Oratrix that includes a SMIL authoring tool.

SOJA Cherbourg 2

<http://www.helio.org/products/smil/>

A Java-based SMIL player available for download.

Authoring Tools

Although SMIL presentations can be hand-coded, software is available that allows for a more intuitive approach to authoring followed an automated output of code. The URLs that follow provide information on most of the tools now available.

RealSlideShow

<http://www.realnetworks.com/products/slideshow/index.html>

This tool provides an intuitive interface for creating RealPlayer-based image presentations timed to music.

Fluition

<http://www.confluenttechnologies.com/>

This tool uses an object-oriented approach to authoring. To take full advantage of this software, some SMIL authoring knowledge is helpful.

GRiNS

<http://www2.oratrix.nl/>

GRiNS is designed with the multimedia artist in mind and is part of the Oratrix SMIL package.

HomeSite (Allaire)

<http://www.allaire.com/products/homesite/index.cfm>

Long respected as an HTML authoring tool, HomeSite can also output SMIL files.

SMIL 2.0 Authoring Resources

This section provides links to articles, tutorials, and other resources for creating SMIL 2.0 presentations.

JustSMIL (Streaming Media World)

<http://smw.internet.com/smil/smilhome.html>

One of the early comprehensive online resources for all things SMIL. One of the authors of this book, coincidentally, is the founder of the JustSMIL site.

HTML+Time 2.0

<http://msdn.microsoft.com/library/default.asp?url=/workshop/author/behaviors/time.asp>

A primer on how to use Microsoft's updated HTML+Time for SMIL presentations in Internet Explorer 5.0 and later.

RealSystem Production Guide: Assembling a Presentation with SMIL 2.0

<http://service.real.com/help/library/guides/production8/realpgd.htm>

The URL listed is a link to the RealSystem Production Guide currently in use. When the new generation of RealPlayer that supports SMIL 2.0 is announced (October 2001), this link should provide access to the updated Production Guide.

The CWI SMIL Page

<http://www.cwi.nl/~media/SMIL/>

Based in the Netherlands, CWI (Centrum voor Wiskunde en Informatica) is the National Research Institute for Mathematics and Computer Science. They offer workshops around the world on implementing SMIL.

Universal SMIL

<http://www.empirenet.com/~joseram/universal/universal.html>

Media formats and content that is playable on all players. A convenient chart.

Webopedia

<http://webopedia.internet.com/TERM/S/SMIL.html>

Contains a brief definition of SMIL and some links.

Web Review Looks at the Future of SMIL

http://www.webreview.com/1999/10_08/designers/10_08_99_1.shtml

An article describing the differences and new features of SMIL 2.0.

SMIL 2.0: Codeless Animation in HTML

<http://hotwired.lycos.com/webmonkey/01/24/index3a.html>

An article by Rodney Reid for Webmonkey on the use of SMIL 2.0 for animation.

SMIL 1.0 Resources

This section provides links to articles, tutorials, and other resources for creating SMIL 1.0 presentations.

JustSMIL (Streaming Media World)

<http://smw.internet.com/smil/smilhome.html>

As listed previously, a comprehensive online resource for all things SMIL, including many articles and tutorials on SMIL 1.0.

A Primer on SMIL

<http://www.webtechniques.com/archives/1998/09/bouthillier/>

Larry Bouthillier, in an article for Web Techniques, provides the background of SMIL's introduction as a W3C standard.

SMIL: Multimedia Markup

<http://hotwired.lycos.com/webmonkey/98/23/index1a.html?tw=authoring>

Another introductory article to the use of SMIL 1.0 for creating multimedia presentations. This article was written by Shvatz for Webmonkey.

An Overview of Using SMIL 1.0

<http://hotwired.lycos.com/webmonkey/00/41/index4a.html?tw=authoring>

Steve McCannell, in an article for Webmonkey, gives an overview of SMIL 1.0.

RealSystem Production Guide: Assembling a Presentation with SMIL 1.0

<http://service.real.com/help/library/guides/production8/realpgd.htm>

Areas of interest in the Production Guide include Chapter 6: "Assembling a Presentation with SMIL 1.0," Chapter 7: "Extending SMIL," Appendix D: "SMIL Quick Reference," and Appendix E: "SMIL Language Codes."

Apple QuickTime's SMIL 1.0 Pages

<http://www.apple.com/quicktime/authoring/qtsmil.html>

<http://www.apple.com/quicktime/authoring/qtsmil2.html>

A little bit of info on how to make use of QuickTime's SMIL extensions.

RealNetwork's SMIL 1.0 Tutorial

<http://www.realnetworks.com/devzone/tutorials/authoring/smil/index.html>

A concise, step-by-step introduction to using SMIL 1.0.

RealNetworks SMIL 1.0 Evaluator

[http://www.realnetworks.com/devzone/howto/
contentcreation/smilevaluator/index.html](http://www.realnetworks.com/devzone/howto/contentcreation/smilevaluator/index.html)

Try this validator to check for common SMIL 1.0 coding errors.

WebReview SMIL 1.0 Articles

http://www.webreview.com/2000/05_05/designers/05_05_00_1.shtml

Follow the links on the bottom of this article to access several additional WebReview articles on SMIL.

Some Additional SMIL 1.0 Online Tutorials

http://www.webreview.com/1999/03_12/designers/03_12_99_1.shtml

<http://www.helio.org/products/smil/tutorial/index.html>

<http://www.empirenet.com/%7Ejoseram/index.html>

Three additional basic introductions to SMIL 1.0.



GLOSSARY

A

active duration The length of time during which the *element* plays normally. If no repeating behavior is specified and an end is not specified, the active duration is the same as the *simple duration*, or the natural duration of the element. If the element is set to play repeatedly, the active duration is defined by how many times the simple duration is repeated.

active time Time measured in relation to the *element's active duration*. Active time is measured from the beginning of the element.

architecture See *closed architecture*, *open architecture*, and *streaming media architecture*.

attributes Code contained within an *element*. In the example `<region id="main">`, *id* is an *attribute* of the *element* *region*. Attributes are generally used to extend or refine an element's function and to provide information about the data, in contrast to being data itself. See also *element*, *value*.

authoring tool A program or application that helps create a hypertext, graphical, or multimedia presentation. Usually provides a graphical interface that allows the user to drag and drop media objects onto a layout and/or into a timeline. The tool then automatically creates the underlying code and/or scripting to implement the user's decisions.

average data rate Represents the average amount of data per time unit that a presentation outputs. Calculated by dividing a presentation's combined file size by its duration.

B

bandwidth The amount of data that can pass through a network connection in a given time period. Usually measured in *kilobits per second (Kbps)* and primarily dependent on the hardware, software, and connection rate of a client machine.

bit The smallest unit of data measurement. Assigned a binary value of 1 or 0. Eight bits are in a *byte*.

bit rate A measurement of bandwidth and also the rate at which a presentation is streamed. Usually expressed as *kilobits per second (Kbps)*. See also *constant bit rate* and *variable bit rate*.

buffering Receiving and storing data before it is played back. Initial buffering of a presentation is called *preroll*, during which files are accessed by the *client* machine. Extensive buffering during a presentation can stall playback entirely.

byte A common data measurement. One byte consists of eight bits.

C

cascading style sheets (CSS) Adds formatting information to *HTML* and *XML* in layers, where each layer overrules the previous one.

child element An element *nested* within another one, with the outer element referred to as the *parent element*.

client An application that receives data from a *server*. For instance, a *RealPlayer* receives data from a *RealServer*, the same as a Web browser receives data from a Web server.

clip A media file within a presentation. Many clips have internal timelines.

clock-value The numeric means of representing time. *Full*, *partial*, or *time-count* clock-values are permitted in SMIL coding.

closed architecture A proprietary system owned by a manufacturer. Usually only products and devices from the manufacturer will function in a closed system. *HTML+Time* is an example of a closed architecture that functions only in the Internet Explorer browser on Windows machines. See also *open architecture*.

codec A compression/decompression software component that translates data between its uncompressed and compressed forms. Reduces the *bandwidth* required by a streaming clip.

constant bit rate (CBR) A method of encoding audio and video files so that the *bit rate* of the file remains the same throughout the entire file. See also *bit rate* and *variable bit rate (VBR)*.

continuous media Media that is naturally time-based and generally supports intrinsic timing and duration (although the duration might be indefinite). These media are sometimes described as “time-based” or “played” media. This includes most audio, video, and time-based animations. See also *discrete media*.

CSS See *cascading style sheets*.

D

data rate The number of *bytes* per second used to represent a media object. Usually measured in *kilobytes* per second (*KBps*) or megabytes per second (*MBps*).

definite time Time resolved to a finite, non-indefinite value. See also *desired time*, *effective time*, *explicit time*, and *implicit time*.

discrete media Media that does not have intrinsic timing or intrinsic duration. This includes images, text, and some vector media. See also *continuous media*.

duration See *active duration* and *simple duration*.

E

effective time Time that is actually observed at document playback. It reflects both the constraints of the timing model as well as real-world issues, such as media delivery. See also *desired time*, *explicit time*, and *implicit time*.

element The most basic unit of code, which acts as a container for data presentation and modification. In the example `<region id="main">`, `<region>` is the element. See also *attribute* and *value*.

encoding Usually found in a *streaming media architecture*, the encoding process translates data into a compressed form that is understood by the architecture’s *player* or *plug-in*.

explicit time A time that has been specified by the author, using the SMIL syntax. See also *implicit time*.

eXtensible Markup Language See *XML*.

F

full clock-value A means to represent a moment in time measured in hours, minutes, and seconds. For example, 01:15:25.12 would be interpreted as 1 hour, 15 minutes, 25 seconds and 12 milliseconds. See also *partial clock-value* and *timecount value*.

H

helper application A program or file that adds functionality to another program. Often browsers use these applications to play back multimedia. Also called a *plug-in*.

host-language conformant profile A SMIL 2.0 *language profile* that incorporates a Structure module, which in turn contains the <smil> element. *Profiles* further enhance the potential for smooth playback of multimedia presentations across *players* and browsers. The main host-language conformant profiles are the *SMIL 2.0 Language Profile* and the *SMIL 2.0 Basic Language Profile*. See also *integration-set conformant profile*.

HTML (Hypertext Markup Language) An authoring language used to define the structure and layout of a Web document.

HTML+Time (HTML+Timed Interactive Multimedia Extensions) A proposal put forth to the W3C by Microsoft at the time *SMIL 1.0* was adopted that extends *HTML* and enables timing and interactivity without using scripting languages. Supported only by the Internet Explorer browser on Windows machines. In *SMIL 2.0*, this approach can be found in the *XHTML+SMIL 2.0 Language Profile*.

HTTP (Hypertext Transfer Protocol) The underlying protocol used by the World Wide Web. Can be used for *streaming* media. See also *TCP*.

Hypertext Markup Language See *HTML*.

I

implicit time Time that is defined intrinsically by the element media (for example, based on the length of a movie) or by the time model semantics (such as duration of a *par* time container). See also *explicit time*.

integration-set conformant profile A SMIL 2.0 *language profile* that does not include the root `<smil>` element, but instead incorporates the SMIL code into an HTML document that is displayed as a Web page. *HTML+Time* is an example of this type of language profile. See also *host-conformant language profile*.

intrinsic duration The natural duration of a media file when it follows its timeline from start to finish.

K

Kbps *Kilobits per second*. A common measurement of *bandwidth*.

KBps *Kilobytes per second*, also referred to as “Kps.” A common measurement of a file’s *data rate*.

kilobit (Kb) 1024 bits.

kilobyte (KB) A common data measurement equal to 1024 bytes.

L

language profile Also called *profiles*, these standards combine modules and present established characteristics for playing back multimedia. The intent behind profiles is an effort to enhance the potential for smooth playback of multimedia presentations across *players* and browsers. Currently, two types of language profiles make use of SMIL 2.0: *host-language conformant profile* and *integration-set conformant profile*.

layout The graphical or visual structure of a document or presentation.

M

media objects The files that make up a multimedia presentation, including audio, video, still image, animation, HTML pages, and/or text files. Called through the use of text-only coding, the links to these objects make up a significant portion of a *SMIL* presentation. See also *Web objects*.

media time Time measured in relation to the object's *duration*. Media time is measured from the beginning of the media, as modified by any `clipBegin` or `clipEnd` attributes.

meta-information *Meta* is a common prefix that means "about." Therefore, meta-information is information about the data being presented and commonly includes information about the author, copyright, keywords, and other indexing and presentation data. In *SMIL 2.0*, the Metainformation module provides the arena for exchanging this information.

MIME types MIME is an acronym for multipurpose Internet mail extensions, which is a specification for formatting non-ASCII messages so that they can be sent over the Internet. Web browsers support various MIME types, which identify the file type for graphics, audio, and video files, among others.

N

nested Embedding one object within another object of the same type. This structure often occurs in programming and code writing when an *element* is nested within another element. See also *child element* and *parent element*.

non-temporal Not related to time or attached to a timeline.

O

open architecture A system with publicly published specifications and standards. Allows the system to connect easily with other devices, programs, and systems that also conform to the published standards. *SMIL* and *HTML* are both examples of open architecture systems. See also *closed architecture*.

P

packets A unit of data sent over the Internet using *HTTP* and *TCP* protocols. Files and documents are broken into packets by *TCP*, which then reassembles the data at the *client* end. If any data is damaged or missing, the packets are re-sent.

parent element When elements are *nested*, the parent element is the outer element. *Child elements* are then *nested* within the parent element.

partial clock-value A means to represent a moment in time in which unused time components are hidden. For example, `begin="00:30.25"` will cause the element to start playing at 30.25 seconds. See also *full clock-value* and *timecount value*.

player The *client* application that plays back multimedia that has been encoded for use by the player. See also *streaming media architecture*.

plug-in A program or file that adds functionality to another program. Often browsers use these applications to play back multimedia. Also called *helper applications*.

preroll The initial *buffering* of a presentation during which files are accessed by the *client* machine.

profiles New in SMIL 2.0, profiles combine modules and present established characteristics for playing back multimedia. The intent behind profiles is an effort to enhance the potential for smooth playback of multimedia presentations across players and browsers. See also *language profile*.

progressive download A file that is transmitted from a standard *HTTP*/Web server using *TCP/IP* and is downloaded to the *client* machine. Playback begins when enough data has been received for the entire file to play back without interruption. Often mistaken for streaming. Also called "pseudo-streaming."

pseudo streaming See *progressive download*.

Q

QuickTime Developed by Apple Computers, QuickTime is an architecture for the creation and delivery of multimedia. The QuickTime architecture provides a popular *player* and *plug-in*, as well as *server* software and an *encoding* device. The QuickTime player also has the capacity to play back many well-known file formats that are not specifically QuickTime files. A robust and well-distributed system for Web multimedia, QuickTime currently supports SMIL 1.0 playback. The next version of the player is expected to support SMIL 2.0. See also *streaming architecture*.

R

RealMedia A blanket term used to refer to the various "Real" data types that can be streamed using a *RealPlayer*. File types referred to as RealMedia include RealAudio and RealVideo (.rm), RealFlash, *RealPix* (.rp), and *RealText* (.rt).

RealPix (.rp) A file type that uses a proprietary markup language to stream still images along with special effects, such as zooms and fades. Requires a *RealPlayer* for viewing.

RealPlayer The player for *RealMedia* files. Supports *SMIL* 1.0 playback and will support *SMIL* 2.0 with the release of the RealONE version of the player.

RealSlideshow A RealNetworks tool that uses *SMIL* 1.0 to create streaming slideshows. See also *streaming architecture*.

RealSystem An *architecture* for the delivery of multimedia developed by RealNetworks, which includes *RealPlayer* (player and *plug-in*), *RealServer* (server software), and *RealProducer* (encoding device).

RealText (.rt) A file type that uses a proprietary markup language to present streaming text. Requires a *RealPlayer* for viewing. See also *streaming architecture*.

real-time Delivery that occurs simultaneously with an event or playback of a file.

region In *SMIL* coding, an area designated for playback of media elements.

RTP (Real-Time Protocol) The Internet standard protocol for the transport of *real-time* data. None of the commercial *streaming* products currently use RTP, but the protocol is expected to be adopted by streaming vendors over time. See also *RTSP*, *TCP*, and *UDP*.

RTSP (Real-Time Streaming Protocol) An open, standards-based control protocol developed by RealNetworks and Netscape. RTSP works with established protocols such as *UDP* and *TCP* to stream multimedia to *RTP*-based client players, including *RealPlayer*. See also *TCP*.

S

Scalable Vector Graphics (SVG) An *XML*-based standard language for vector graphics recommended by the *W3C*.

server A computer that other computers connect to for the purpose of retrieving information. The machines that are connecting to the server are called *client* machines.

simple duration The *duration* defined by the basic `begin` and `dur` markup. It does not include any of the effects of playing repeatedly or of `fill`. If the *explicit times* are not specified, the simple duration is defined as the element's implicit duration. See also *active duration*.

simple time Time measured in relation to the element's *simple duration*. Simple time is measured as an offset from the beginning of a particular instance of the *simple duration*.

SMIL (Synchronized Multimedia Integration Language) A simple but powerful XML-compliant markup language that coordinates how and when multimedia files play back on *client* machines. Created using a *text editor* or any other software that saves output as plain text with line breaks.

SMIL 2.0 Basic Language Profile A stripped-down version of the *SMIL 2.0 Language Profile*, intended for use with personal digital devices, mobile phones, and entertainment devices. This language profile is also a *host-conformant language profile* currently making use of the SMIL 2.0 modules.

SMIL 2.0 Language Profile The main *host-conformant language profile* currently making use of the SMIL 2.0 modules. Incorporates a Structure module and the use of the `<smil>` element.

SMIL Boston The name given to the working draft of *SMIL 2.0*.

specification (W3C) A publicly available recommendation made by the *World Wide Web Consortium* that describes the building blocks of the Web. See also *W3C*.

spline A smooth curve that passes through two or more points.

streaming A technique for transferring data as a steady and continuous stream that allows data to be viewed before the entire file is transmitted. Also used to refer to transferring data in *real-time*.

streaming media architecture A system for delivering multimedia over the Web. Usually an architecture includes server software, an *encoding* device, and a *player* or *helper application*. *QuickTime*, *RealNetworks*, and *Windows Media* are all streaming media architectures. See also *streaming*.

syntax The spelling and grammar of a programming language.

T

TCP (Transmission Control Protocol) A protocol used by *HTTP* to ensure reliable document transfer. Before sending, TCP breaks the data into *packets*, which are then reassembled at the *client* end. If any data is missing or damaged, the entire document is re-sent. See also *RTSP* and *UDP*.

temporal Having to do with time and timelines.

text editor An application that enables writing and saving plain text files with line breaks. Used to create *HTML* and *SMIL* documents “by hand.” Notepad (Windows) and SimpleText (Macintosh) are examples of text editors.

timecount value A means to represent a moment in time in which time divisions are abbreviated. Using timecount values, the moment in time of 1 hour, 30 minutes, and 15 seconds becomes “1h30m15s.” See also *full clock-value* and *partial clock-value*.

true streaming See *streaming*.

U

UDP (User Datagram Protocol) A protocol for the delivery of media over the Web, and the alternative to *TCP*. UDP forsakes TCP’s error correction and allows data dropout to maintain real-time delivery of a presentation. The best choice for live broadcasts because all viewers get the same information simultaneously. A downside of UDP is that the transmissions are often blocked by network firewalls.

URI (Uniform Resource Identifier) The generic term for an absolute or relative address of a resource. When the address given is absolute, as in `http://www.example.com`, the first part of the address indicates what protocol to use, and the second part specifies the IP address or the domain name where the resource is located. When the address is relative, as in `page.html`, the address indicates the path to the resource relative to the document in which it appears. A URL is one kind of URI.

V

value In coding, the data assigned to an *attribute* within an *element*. In the example `<region id="main">`, the element is `<region>`, `id` is the attribute, and `main` is the value. Values should always be enclosed in quotation marks.

variable bit rate (VBR) A method of *encoding* that changes the bit rate over the course of the file. This allows the encoder to allocate more bits to areas that are harder to encode and fewer bits to areas that are easier, which creates higher-quality compressed files. See also *bit rate* and *constant bit rate (CBR)*.

W

W3C (World Wide Web Consortium) A group of researchers and industry representatives who develop *specifications*, guidelines, software, and tools that are universally adaptable for the Web. One particular task of the W3C is to produce *specifications*, or recommendations, that describe building blocks for the Web. The *SMIL* specifications are examples of these recommendations that are public and available to all.

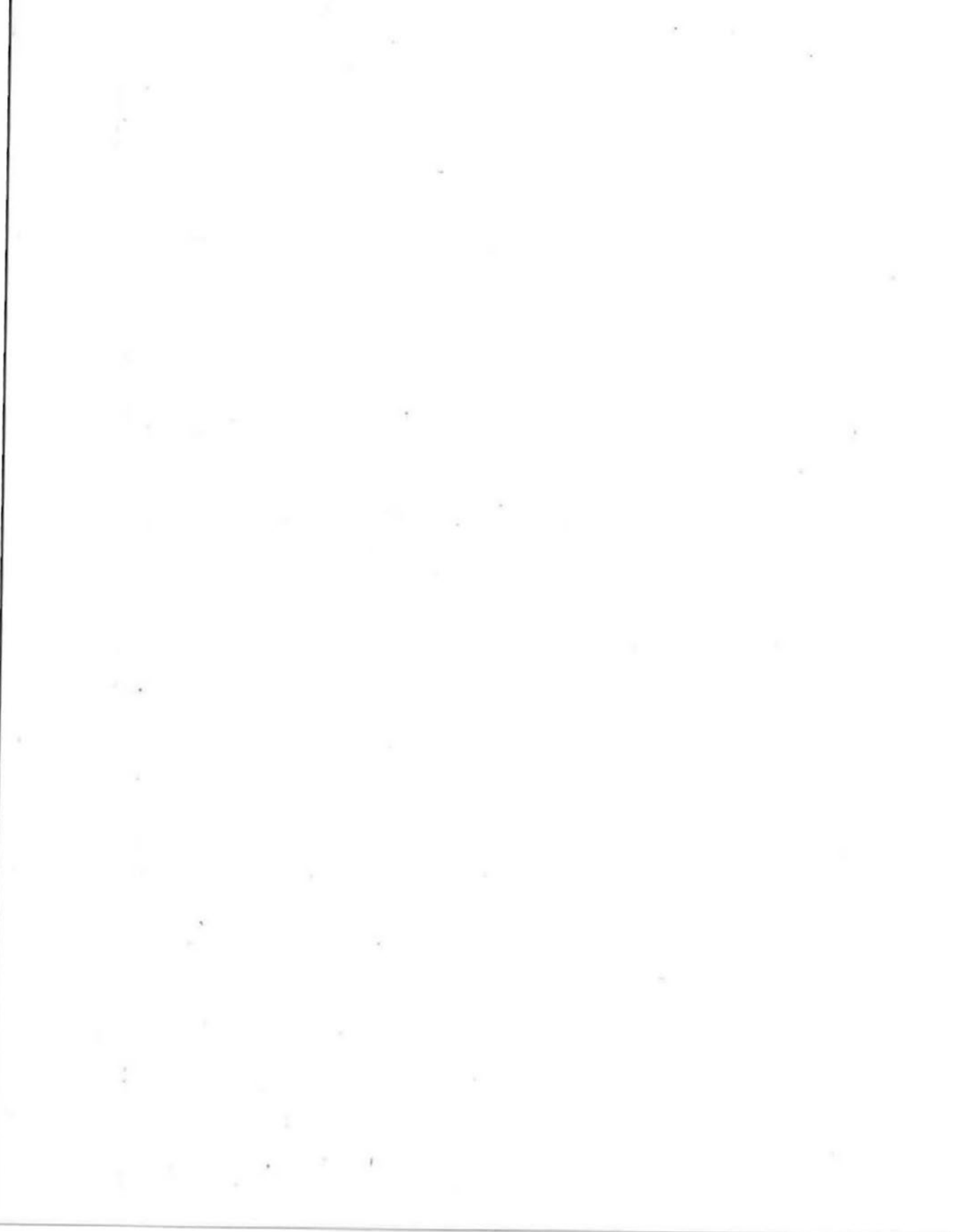
Web objects The online files used in a multimedia presentation, including Web sites and links to audio, video, still image, animation, and/or text files on the Web. Called into action through the use of text-only coding, the links to these objects make up a significant portion of a *SMIL* presentation. See also *media objects*.

Windows Media Player Formerly called NetShow. An architecture for the delivery of multimedia using the Internet Explorer browser. Still in development for use on the Macintosh OS. See also *streaming architecture*.

X

XHTML+SMIL 2.0 Language Profile A *language profile* that is *integration-set conformant* rather than *host-language conformant* and makes use of the *SMIL* 2.0 modules. This profile does not include the root `<smil>` element; instead, it incorporates the *SMIL* code into the Web page along with *HTML*.

XML (eXtensible Markup Language) Separates content from formatting. Allows for the creation of customized element sets to provide functionality not available with *HTML*. XML is a language for defining languages. *SMIL* is an XML-compliant markup language.



Index

A

- abstract attribute (Media Object modules), 120
- accelerate attribute (Time Manipulations module), 230-231
- accesskey attribute (Linking modules), 194
- accumulate attribute (Animation modules), 171-173
- actuate attribute
 - Animation modules, 165
 - Linking modules, 195
- adding
 - logos (RealPix), 296
 - timing, HTML+TIME presentations, 313-316
- additive attribute (Animation modules), 173-175
- alt attribute
 - Linking modules, 195
 - Media Object modules, 120-121
- animate element
 - Animation modules, 159
 - RealPix, 241
- animateColor element (Animation modules), 161-162
- animateMotion element (Animation modules), 160
- animation element (Media Object modules), 117
- Animation modules, 44, 157
 - attributes, 162, 352-353
 - accumulate, 171-173
 - actuate, 165
 - additive, 173-175
 - attributeName, 162-163
 - attributeType, 163
 - by, 168-169
 - calcMode, 170-171
 - from, 166-167
 - href, 164

- keySplines, 178-187
- keyTimes, 177-178
- origin, 175
- path, 175-176
- show, 165
- targetElement, 163
- to, 167
- type, 165
- values, 169
- BasicAnimation, 158-187
 - elements, 158, 352
 - animate, 159
 - animateColor, 161-162
 - animateMotion, 160
 - set, 159-160
 - SplineAnimation, 158-187
- Apple Computer, QuickTime. *See* QuickTime
- area element (Linking modules), 193-194
- aspect attribute (RealPix), 244
- attributeName attribute (Animation modules), 162-163
- attributes. *See also* tags
 - Animation modules (BasicAnimation and SplineAnimation), 162-187, 352-353
 - accumulate, 171-173
 - actuate, 165
 - additive, 173-175
 - attributeName, 162-163
 - attributeType, 163
 - by, 168-169
 - calcMode, 170-171
 - from, 166-167
 - href, 164
 - keySplines, 178-187
 - keyTimes, 177-178
 - origin, 175
 - path, 175-176
 - show, 165
 - targetElement, 163
 - to, 167
 - type, 165
 - values, 169
 - BasicContentControl module
 - system-overdub-or-caption, 217
 - systemAudioDesc, 212
 - systemBitrate, 213
 - systemCaptions, 214
 - systemComponent, 214
 - systemCPU, 215
 - systemLanguage, 215-216
 - systemOperatingSystem, 216
 - systemOverdubOrSubtitle, 217
 - systemRequired, 218
 - systemScreenDepth, 218
 - systemScreenSize, 219
 - Content Control modules, 354-355
 - CustomTestAttributes module
 - customTest, 221
 - defaultState, 222
 - override, 223
 - uid, 223
 - Layout modules, 70, 346-347
 - background-color, 71
 - backgroundColor, 70
 - bottom, 71-72
 - close, 72-73
 - fit, 73
 - height, 74-75
 - id, 74
 - left, 71-72
 - open, 75
 - regAlign, 75-76
 - region, 78
 - regionName, 78
 - regPoint, 76-77
 - right, 71-72
 - showBackground, 78
 - soundLevel, 79
 - top, 71-72
 - type, 79
 - width, 74-75
 - z-index, 80-81
 - Linking modules, 194, 353-354
 - accesskey, 194
 - actuate, 195
 - alt, 195
 - coords, 196-197
 - destinationLevel, 198-199
 - destinationPlaystate, 199
 - external, 199
 - fragment, 200
 - href, 201
 - nohref, 202
 - shape, 202-203
 - show, 203
 - sourceLevel, 203
 - sourcePlaystate, 204
 - tabindex, 204-205
 - target, 205-206
 - Media Object modules, 120, 349-350
 - abstract, 120
 - alt, 120-121

Index

A

- abstract attribute (Media Object modules), 120
- accelerate attribute (Time Manipulations module), 230-231
- accesskey attribute (Linking modules), 194
- accumulate attribute (Animation modules), 171-173
- actuate attribute
 - Animation modules, 165
 - Linking modules, 195
- adding
 - logos (RealPix), 296
 - timing, HTML+TIME presentations, 313-316
- additive attribute (Animation modules), 173-175
- alt attribute
 - Linking modules, 195
 - Media Object modules, 120-121
- animate element
 - Animation modules, 159
 - RealPix, 241
- animateColor element (Animation modules), 161-162
- animateMotion element (Animation modules), 160
- animation element (Media Object modules), 117
- Animation modules, 44, 157
 - attributes, 162, 352-353
 - accumulate, 171-173
 - actuate, 165
 - additive, 173-175
 - attributeName, 162-163
 - attributeType, 163
 - by, 168-169
 - calcMode, 170-171
 - from, 166-167
 - href, 164

Time Manipulations module, 228, 355

- accelerate, 230-231
- autoReverse, 232-233
- decelerate, 231-232
- speed, 228-229

Timing and Synchronization module, 93, 348-349

- begin, 93-98
- dur, 98-99
- end, 93-98
- endsync, 100
- fill, 102
- fillDefault, 103
- higher, 103-104
- lower, 104-105
- max, 99
- min, 99
- pauseDisplay, 105-106
- peers, 106-107
- repeat, 100-101
- repeatCount, 101
- repeatDur, 101
- restart, 107
- restartDefault, 108
- syncBehavior, 108-109
- syncBehaviorDefault, 109
- syncMaster, 111
- syncTolerance, 109-110
- syncToleranceDefault, 110
- timeAction, 112-114
- timeContainer, 111-112

Transition Effects module, 138, 350-352

- begin, 139
- borderColor, 139
- borderWidth, 139
- by, 140
- calcMode, 140
- direction, 141
- dur, 142
- end, 139
- endProgress, 142
- fadeColor, 142
- from, 143
- horzRepeat, 143
- href, 143
- id, 145
- mode, 145
- repeatCount, 145
- repeatDur, 145
- startProgress, 142
- subtype, 149

- targetElement, 146
- to, 143
- transIn, 146-148
- transOut, 146-148
- type, 149
- values, 149-150
- vertRepeat, 143-144

attributeType attribute (Animation modules), 163

audio element (Media Object modules), 117

AudioLayout module, 64

author attribute

- Media Object modules, 121
- RealPix, 244

authoring with SMIL, 19

- bandwidth, 21-24
 - calculating, 24-25
 - compensating for bandwidth, 26
- browser compliance and compatibility, 27
- data rates, 21-24
 - calculating, 24-25
- document structure, 20-21
- resources, 359-361
- rules, 19-20
- serving your work, 27-28
 - streaming architectures, 28-30
- tools, 31
 - Fluition, 32-33
 - GRiNS, 32
 - HomeSite, 35
 - Macromedia Flash, 34-35
 - RealSlideshow, 33-34

autoReverse attribute (Time Manipulations module), 232-233

B

b element (RealText), 260

backgroundColor attribute

- Layout module, 70-71
- RealPix, 245
- subregion positioning, 82

backgrounds, HTML+TIME presentations, 331-332

bandwidth, 21-24

- calculating, 24-25
- compensating for bandwidth, 26
- connection speed, 23
- HTML+TIME presentations, 333-335
- RealPlayer formats, 237

bandwidth attribute (PrefetchControl module), 225

basic document, 7-8

BasicAnimation (Animation module), 158-187

BasicContentControl module, 207-219

attributes

- system-overdub-or-caption, 217
- systemAudioDesc, 212
- systemBitrate, 213
- systemCaptions, 214
- systemComponent, 214
- systemCPU, 215
- systemLanguage, 215-216
- systemOperatingSystem, 216
- systemOverdubOrSubtitle, 217
- systemRequired, 218
- systemScreenDepth, 218
- systemScreenSize, 219

elements, 209

- switch, 210-212

test attributes, 212

BasicLayout module, 64

begin attribute

- RealText, 263-264
- Timing and Synchronization module, 93-98
- Transition Effects module, 139

bgcolor attribute (RealText), 264

bitrate attribute (RealPix), 245-246

body element (Structure module), 53-54

body section (SMIL document structure), 21

borderColor attribute (Transition Effects module), 139

borderWidth attribute (Transition Effects module), 139

bottom attribute

- Layout module, 71-72
- subregion positioning, 82

br element (RealText), 260

browser compliance and compatibility, 27

browsers and players, 10-16

- GRiNS player, 16
- Internet Explorer, 14-16
- QuickTime, 13-14
- RealPlayer, 10-13
- SOJA, 16-17

brush element

- Media Object modules, 118
- RealONE Player, painting the screen, 281-282

buffering, 26

buttons, labeling (RealPix), 295-296

by attribute

- Animation modules, 168-169
- Transition Effects module, 140

C

calcMode attribute

- Animation modules, 170-171
- Transition Effects module, 140

calculating bandwidth and data rates, 24-25

Cascading Style Sheets. *See* CSS

center element (RealText), 261

charset attribute (RealText), 264

CHECKSUM (TCP/IP protocol), 29

class attribute (Structure module), 54

clear element (RealText), 258

clipBegin attribute (Media Object modules), 121-123

clipEnd attribute (Media Object modules), 121-123

close attribute (Layout module), 72-73

coding SMIL

authoring tools, 31

- Fluition, 32-33
- GRiNS, 32
- HomeSite, 35
- Macromedia Flash, 34-35
- RealSlideshow, 33-34

by hand, 19

- bandwidth, 21-26
- browser compliance and compatibility, 27
- data rates, 21-25
- document structure, 20-21
- rules, 19-20
- serving your work, 27-30

color attribute

- Media Object modules, 123
- RealPix, 247
- RealText, 264-265

command attribute (RealText), 265

compatibility

- browser, 27
- RealPix, 236-237
- RealText, 236-237

compliance, browsers, 27

Confluent Technologies Web site, 33

conformance profiles

- Content Control modules, 208-209
- Layout module, 64
- Linking modules, 190
- Media Object modules, 116
- Structure module, 49-50
- Time Manipulations module, 227
- Timing and Synchronization module, 86
- Transition Effects module, 136

connection speed, bandwidth, 23

content attribute (Metainformation module), 61

- Content Control modules, 44, 207-208
 - attributes, 354-355
- BasicContentControl module, 207-209
 - attributes, 212-219
 - elements, 209-212
 - test attributes, 212
- conformance profiles, 208-209
- CustomTestAttributes module, 207, 220-223
 - attributes, 221-223
 - elements, 220-221
- elements, 354
- PrefetchControl module, 207, 224-225
 - attributes, 225
 - elements, 224
- SkipContentControl module, 207
 - attributes, 226
- coords attribute (Linking modules), 196-197
- copyright attribute
 - Media Object modules, 123
 - RealPix, 247
- crawlrate attribute (RealText), 265
- creating
 - layout, 279-281
 - RealText, 296-297
- crossfade element (RealPix), 241
- crossfades (RealPix), 287-288
- CSS (Cascading Style Sheets), 158, 310
- customAttributes element (CustomTestAttributes module), 220
- customTest
 - attribute (CustomTestAttributes module), 221
 - element (CustomTestAttributes module), 221
- CustomTestAttributes module, 207, 220-223
 - attributes
 - customTest, 221
 - defaultState, 222
 - override, 223
 - uid, 223
 - elements, 220
 - customAttributes, 220
 - customTest, 221

D

- data rates, 21-24
 - calculating, 24-25
- decelerate attribute (Time Manipulations module), 231-232
- defaultState attribute (CustomTestAttributes module), 222

- delivery of presentations, 27
 - streaming architectures, 28
 - pseudostreaming, 28-29
 - true streaming, 29-30
- depreciation, SMIL 2.0, 46
- destinationLevel attribute (Linking modules), 198-199
- destinationPlaystate attribute (Linking modules), 199
- direction attribute
 - RealPix, 247-248
 - Transition Effects module, 141
- document structure (authoring SMIL by hand), 20-21
- dsth attribute (RealPix), 248-249
- dstw attribute (RealPix), 248-249
- dstx attribute (RealPix), 248-249
- dsty attribute (RealPix), 248-249
- dur attribute
 - Timing and Synchronization module, 98-99
 - Transition Effects module, 142
- duration attribute
 - RealPix, 249
 - RealText, 265

E

- elements
 - Animation modules (BasicAnimation and SplineAnimation), 158, 352
 - animate, 159
 - animateColor, 161-162
 - animateMotion, 160
 - set, 159-160
 - BasicContentControl module, 209
 - switch, 210-212
 - Content Control modules, 354
 - CustomTestAttributes module, 220
 - customAttributes, 220
 - customTest, 221
 - Layout modules, 64, 346
 - layout, 65-66
 - region, 66-67
 - regPoint, 67-69
 - root-layout, 69
 - topLayout, 69-70
 - Linking modules, 190-193, 353
 - a, 191
 - area, 193-194
 - Media Object modules, 116, 349
 - animation, 117
 - audio, 117

- brush, 118
 - img, 117
 - param, 118-119
 - ref, 117
 - text, 117
 - textstream, 117
 - video, 117
 - Metainformation module, 57, 346
 - meta, 58
 - metadata, 58-59
 - PrefetchControl module, prefetch, 224
 - RealPix, 237
 - animate, 241
 - crossfade, 241
 - fadein, 240
 - fadeout, 240
 - fill, 240
 - head, 239
 - image, 239-240
 - imfl, 238
 - viewchange, 242-243
 - wipe, 241
 - RealText, 257
 - b, 260
 - br, 260
 - center, 261
 - clear, 258
 - font, 263
 - hr, 261
 - href, 263
 - i, 261
 - li, 261
 - ol, 262
 - p, 262
 - pos, 259
 - pre, 262
 - required, 259
 - s, 262
 - time, 259
 - tl, 260
 - tu, 260
 - u, 262
 - ul, 263
 - window, 257-258
 - Structure module, 50, 345
 - body, 53-54
 - head, 51-52
 - Timing and Synchronization module, 86-92, 347
 - excl, 90-92
 - par, 88-90
 - priorityClass, 92
 - seq, 86-88
 - Transition Effects module, 136, 350
 - transition, 137-138
 - transitionFilter, 138
 - end attribute
 - RealText, 263-264
 - Timing and Synchronization module, 93-98
 - Transition Effects module, 139
 - endProgress attribute (Transition Effects module), 142
 - endsync attribute (Timing and Synchronization module), 100
 - erase attribute (Media Object modules), 124
 - excl element (Timing and Synchronization module), 90-92
 - exclusive interactivity with SMIL 2.0, 292-295
 - external attribute (Linking modules), 199
 - extraspaces attribute (RealText), 266
- ## F
- face attribute (RealText), 266
 - fadeColor attribute (Transition Effects module), 142
 - fadein element (RealPix), 240
 - fadeout element (RealPix), 240
 - fades, RealPix, 287-288
 - fill attribute (Timing and Synchronization module), 102
 - fill element (RealPix), 240
 - fillDefault attribute (Timing and Synchronization module), 103
 - final presentation, HTML+TIME presentations, 337, 342
 - fit attribute
 - Layout module, 73
 - subregion positioning, 82
 - Flash animation, 281
 - Fluition, 32-33
 - font element (RealText), 263
 - fragment attribute (Linking modules), 200
 - from attribute
 - Animation modules, 166-167
 - Transition Effects module, 143
 - from-to animation, 166
 - future of SMIL, 17
- ## G-H
- goals
 - SMIL 1.0, 40
 - SMIL 2.0, 44

graphics, putting in motion, HTML+TIME presentations, 329-330
GRiNS, 16, 32

hand coding SMIL, 19
 bandwidth, 21-24
 calculating, 24-25
 compensating for bandwidth, 26
 browser compliance and compatibility, 27
 data rates, 21-24
 calculating, 24-25
 document structure, 20-21
 rules, 19-20
 serving your work, 27-28
 streaming architectures, 28-30
handle attribute (RealPix), 249-250
handles (elements), 21
head element
 RealPix, 239
 Structure module, 51-52
head section (SMIL document structure), 21
height attribute
 Layout module, 74-75
 RealPix, 250
 RealText, 266
 subregion positioning, 82
Helio SOJA. *See* SOJA
HierarchicalLayout module, 64
higher attribute (Timing and Synchronization module), 103-104
HomeSite, 35
horzRepeat attribute (Transition Effects module), 143
hr element (RealText), 261
href attribute
 Animation modules, 164
 Linking modules, 201
 RealText, 263
 Transition Effects module, 143
HTML+TIME presentations, applying SMIL to
 Internet Explorer
 adding timing, 313-316
 bandwidth, 333-335
 building the background, 331-332
 final presentation, 337-342
 matching the planned layout, 324-329
 planning the presentation, 308-309
 putting graphics in motion, 329-330
 roughing out the page, 309-310
 stalling, 335-337
 structuring the presentation, 311-313
 templates, 310-311

time containers, 316-318
transitions, 318-324

I
i element (RealText), 261
id attribute
 Layout module, 74
 Metainformation module, 61
 Structure module, 54
 Transition Effects module, 145
image element (RealPix), 239-240
images, loading into RealPix, 287
imfl element (RealPix), 238
img element (Media Object modules), 117
Internet Explorer (applying SMIL), 14-16, 305-306
 cross-platform Web sites, 306
 HTML+TIME presentations
 adding timing, 313-316
 bandwidth, 333-335
 building the background, 331-332
 final presentation, 337-342
 matching the planned layout, 324-329
 planning the presentation, 308-309
 putting graphics in motion, 329-330
 roughing out the page, 309-310
 stalling, 335-337
 structuring the presentation, 311-313
 templates, 310-311
 time containers, 316-318
 transitions, 318-324
 tutorial, 307-308
 Web browser compatibility, 306
interpolation, linear, 170

J-K

Kb (kilobits), bandwidth measurement, 24
KB (kilobytes), data rates measurement, 24
keySplines attribute (Animation modules), 178-187
keyTimes attribute (Animation modules), 177-178

L

labeling buttons (RealPix), 295-296
layout, creating, 279-281
layout element (Layout module), 65-66
Layout modules, 44, 63-63
 attributes, 70, 346-347
 background-color, 71
 backgroundColor, 70

- bottom, 71-72
 - close, 72-73
 - fit, 73
 - height, 74-75
 - id, 74
 - left, 71-72
 - open, 75
 - regAlign, 75-76
 - region, 78
 - regionName, 78
 - regPoint, 76-77
 - right, 71-72
 - showBackground, 78
 - soundLevel, 79
 - top, 71-72
 - type, 79
 - width, 74-75
 - z-index, 80-81
 - AudioLayout, 64
 - BasicLayout, 64
 - conformance profiles, 64
 - elements, 64, 346
 - layout, 65-66
 - region, 66-67
 - regPoint, 67-69
 - root-layout, 69
 - topLayout, 69-70
 - HierarchicalLayout, 64
 - MultiWindowLayout, 64
 - layout section (SMIL document structure), 21
 - left attribute
 - Layout module, 71-72
 - subregion positioning, 82
 - li element (RealText), 261
 - limitations of SMIL 1.0, 43
 - linear interpolation, 170
 - link attribute (RealText), 267
 - linking abilities, SMIL 1.0, 42
 - Linking modules, 44, 189-206
 - attributes, 194, 353-354
 - accesskey, 194
 - actuate, 195
 - alt, 195
 - coords, 196-197
 - destinationLevel, 198-199
 - destinationPlaystate, 199
 - external, 199
 - fragment, 200
 - href, 201
 - nohref, 202
 - shape, 202-203
 - show, 203
 - sourceLevel, 203
 - sourcePlaystate, 204
 - tabindex, 204-205
 - target, 205-206
 - conformance profiles, 190
 - elements, 190, 353
 - a, 191
 - area, 193-194
 - loading images into RealPix, 287
 - logos, adding (RealPix), 296
 - longdesc attribute (Media Object modules), 124
 - loop attribute (RealText), 267
 - lower attribute (Timing and Synchronization module), 104-105
- ## M
- Macromedia Flash, 34-35
 - mailto attribute (RealText), 267
 - matching the planned layout, HTML+TIME presentations, 324-329
 - max attribute (Timing and Synchronization module), 99
 - maxfps attribute (RealPix), 250-251
 - media layout, SMIL 1.0, 41-42
 - Media Object modules, 44, 115-131
 - attributes, 120, 349-350
 - abstract, 120
 - alt, 120-121
 - author, 121
 - clip-begin, 121-123
 - clipBegin, 121-123
 - clip-end, 121-123
 - clipEnd, 121-123
 - color, 123
 - copyright, 123
 - erase, 124
 - longdesc, 124
 - mediaRepeat, 125
 - name, 125
 - readIndex, 126
 - sensitivity, 127-128
 - src, 126
 - title, 128
 - type, 128-129
 - value, 129-130
 - valuetype, 130-131
 - xml:lang, 131
 - conformance profiles, 116

- elements, 116, 349
 - animation, 117
 - audio, 117
 - brush, 118
 - img, 117
 - param, 118-119
 - ref, 117
 - text, 117
 - textstream, 117
 - video, 117
 - mediaRepeat attribute (Media Object modules), 125
 - mediaSize attribute (PrefetchControl module), 225
 - mediaTime attribute (PrefetchControl module), 225
 - meta element (Metainformation module), 21, 58
 - metadata element (Metainformation module), 58-59
 - metainformation, 279
 - Metainformation module, 44, 57
 - attributes, 59, 346
 - content, 61
 - id, 61
 - name, 59-60
 - skip-content, 61
 - elements, 57, 346
 - meta, 58
 - metadata, 58-59
 - mime attribute (RealPix), 251
 - min attribute (Timing and Synchronization module), 99
 - mode attribute (Transition Effects module), 145
 - modules
 - animation, 157
 - attributes, 162-187
 - BasicAnimation, 158-187
 - elements, 158-162
 - SplineAnimation, 158-187
 - BasicContentControl. *See* BasicContentControl module
 - BasicContentControl module
 - Content Control. *See* Content Control modules
 - CustomTestAttributes. *See* CustomTestAttributes module
 - CustomTestAttributes module
 - Layout. *See* Layout modules
 - Linking. *See* Linking modules
 - Media Object. *See* Media Object modules
 - Metainformation. *See* Metainformation module
 - PrefetchControl. *See* PrefetchControl module
 - SkipContentControl. *See* SkipContentControl module
 - SkipContentControl module
 - Structure. *See* Structure module
 - Time Manipulations. *See* Time Manipulations module
 - Timing and Synchronization. *See* Timing and Synchronization module
 - Transition Effects. *See* Transition Effects module
 - motion, putting graphics in motion, HTML+TIME presentations, 329-330
 - multiple SMIL worlds, 46-47
 - MultiWindowLayout module, 64
- ## N-O
- name attribute
 - Media Object modules, 125
 - Metainformation module, 59-60
 - RealPix, 251-252
 - namespace declaration, 278-279
 - nohref attribute (Linking modules), 202
 - ol element (RealText), 262
 - open attribute (Layout module), 75
 - Oratrix Web site, 32
 - origin attribute (Animation modules), 175
 - override attribute (CustomTestAttributes module), 223
- ## P
- p element (RealText), 262
 - painting, brush element, 281-282
 - par
 - element (Timing and Synchronization module), 88-90
 - parallel tag, 21
 - tag, 40
 - param element (Media Object modules), 118-119
 - path attribute (Animation modules), 175-176
 - pauseDisplay attribute (Timing and Synchronization module), 105-106
 - peers attribute (Timing and Synchronization module), 106-107
 - PICS (Platform for Internet Content Selection system), 59
 - planning the presentation (HTML+TIME presentations), 308-309
 - Platform for Internet Content Selection system. *See* PICS
 - players and browsers, 10-16
 - GRiNS player, 16
 - Internet Explorer, 14-16
 - QuickTime, 13-14
 - RealPlayer, 10-13
 - SOJA, 16-17
 - pos element (RealText), 259

- pre element (RealText), 262
 - PrefetchControl module, 207, 224
 - attributes
 - bandwidth, 225
 - mediaSize, 225
 - mediaTime, 225
 - elements, prefetch, 224
 - prefetch element (PrefetchControl module), 224
 - preroll, 26
 - preroll attribute (RealPix), 252
 - priorityClass element (Timing and Synchronization module), 92
 - profiles
 - SMIL 2.0, 45
 - SMIL 2.0 Basic Language Profile, 45
 - SMIL 2.0 Language Profile, 45
 - XHTML+SMIL 2.0 Language Profile, 45
 - Structure module, 49-50
 - Time Manipulations module, 227
 - protocol:path attribute (RealText), 267-268
 - pseudostreaming, 28-29
 - putting graphics in motion, HTML+TIME presentations, 329-330
- ## Q-R
- QuickTime, 13-14
 - RAM files, RealMedia, 301
 - RAMGEN, 301-302
 - readIndex attribute (Media Object modules), 126
 - RealMedia, 300
 - RAM files, 301
 - RealNetworks
 - RealPix, 236-257
 - attributes, 244-257
 - compatibility, 236-237
 - elements, 237-243
 - RealText, 236, 257-270
 - attributes, 263-270
 - compatibility, 236-237
 - elements, 257-263
 - Web site, 34
 - RealONE Player, 274-275
 - adding transition, 282-285
 - creating the layout, 279-281
 - final SMIL 2.0 markup, 302
 - Flash animation, 281
 - metainformation, 279
 - namespace declaration, 278-279
 - painting the screen with the brush element, 281-282
 - player revision, 276
 - tutorial goals, 276-277
 - visual plan for "Villa Travel", 277-278
 - RealPix, 236-257, 270-274
 - adding logos, 296
 - adding video, 290-291
 - attributes, 244
 - aspect, 244
 - author, 244
 - background-color, 245
 - bitrate, 245-246
 - color, 247
 - copyright, 247
 - direction, 247-248
 - dsth, 248-249
 - dstw, 248-249
 - dstx, 248-249
 - dsty, 248-249
 - duration, 249
 - handle, 249-250
 - height, 250
 - maxfps, 250-251
 - mime, 251
 - name, 251-252
 - preroll, 252
 - size, 252
 - srch, 252-253
 - srcw, 252-253
 - srcx, 252-253
 - srcy, 252-253
 - start, 253-254
 - target, 254
 - timeformat, 255
 - title, 255
 - type, 255-256
 - url, 256-257
 - width, 250
 - bringing regions to the top, 295
 - building a G2-capable RealPix file, 286-287
 - compatibility, 236-237
 - elements, 237
 - animate, 241
 - crossfade, 241
 - fadein, 240
 - fadeout, 240
 - fill, 240
 - head, 239
 - image, 239-240
 - imfl, 238
 - viewchange, 242-243
 - wipe, 241

- fades and crossfades, 287-288
- final text file, 288-289
- interactivity with exclusive, 292-295
- labeling buttons, 295-296
- loading images, 287
- referencing in SMIL, 289-290
- replacement with RealONE Player authors, 291-292
- RealPlayer, 10-13
 - RealONE Player, 274-275
 - adding transition, 282-285
 - creating the layout, 279-281
 - final SMIL 2.0 markup, 302
 - Flash animation, 281
 - metainformation, 279
 - namespace declaration, 278-279
 - painting the screen with the brush element, 281-282
 - player revision, 276
 - tutorial goals, 276-277
 - visual plan for "Villa Travel", 277-278
 - RealPlayer G2, 236, 273
 - RealPix, 274, 286-296
 - RealText, 274, 296-297
- RealServers, RAMGEN, 301-302
- RealSlideshow, 33-34
- RealText, 236, 270-274
 - attributes, 263
 - begin, 263-264
 - bgcolor, 264
 - charset, 264
 - color, 264-265
 - command, 265
 - crawlrate, 265
 - duration, 265
 - end, 263-264
 - extraspaces, 266
 - face, 266
 - height, 266
 - link, 267
 - loop, 267
 - mailto, 267
 - protocol:path, 267-268
 - scrollrate, 265
 - size, 268
 - target, 268
 - type, 268-269
 - underline_hyperlinks, 269
 - version, 269
 - width, 266
 - wordwrap, 269
 - x, 270
 - y, 270
 - compatibility, 236-237
 - creating, 296-297
 - elements, 257
 - b, 260
 - br, 260
 - center, 261
 - clear, 258
 - font, 263
 - hr, 261
 - href, 263
 - i, 261
 - li, 261
 - ol, 262
 - p, 262
 - pos, 259
 - pre, 262
 - required, 259
 - s, 262
 - time, 259
 - tl, 260
 - tu, 260
 - u, 262
 - ul, 263
 - window, 257-258
 - referencing in the presentation, 297
- reasons to use SMIL, 8-10
- ref element (Media Object modules), 117
- referencing
 - RealPix in SMIL, 289-290
 - text, RealText, 297
- regAlign attribute (Layout module), 75-76
- region attribute (Layout module), 78
- region element, 41
 - Layout module, 66-67
- regionName attribute (Layout module), 78
- regPoint attribute (Layout module), 76-77
- regPoint element (Layout module), 67-69
- repeat attribute (Timing and Synchronization module), 100-101
- repeatCount attribute
 - Timing and Synchronization module, 101
 - Transition Effects module, 145
- repeatDur attribute
 - Timing and Synchronization module, 101
 - Transition Effects module, 145
- required element (RealText), 259
- resources, Web sites, 357-359
 - authoring resources, 359-361
 - SMIL 1.0, 361-362

restart attribute (Timing and Synchronization module), 107
 restartDefault attribute (Timing and Synchronization module), 108
 right attribute
 Layout module, 71-72
 subregion positioning, 82
 root-layout element (Layout module), 69
 roughing out the page (HTML+TIME presentations), 309-310
 rules, authoring, 19-20

S

s element (RealText), 262
 Scalable Vector Graphics. *See* SVG
 scrollrate attribute (RealText), 265
 sensitivity attribute (Media Object modules), 127-128
 serving your work, 27-28
 streaming architectures, 28
 pseudostreaming, 28-29
 true streaming, 29-30
 seq element (Timing and Synchronization module), 86-88
 set element (Animation modules), 159-160
 shape attribute (Linking modules), 202-203
 show attribute
 Linking modules, 203
 Animation modules, 165
 showBackground attribute (Layout module), 78
 size attribute
 RealPix, 252
 RealText, 268
 skip-content attribute
 Metainformation module, 61
 SkipContentControl module, 226
 SkipContentControl module, 207
 attributes, skip-content, 226
 SMIL (Synchronized Multimedia Integration Language), 7
 application with Internet Explorer, 305-306
 cross-platform Web sites, 306
 HTML+TIME presentations, 308-342
 tutorial, 307-308
 Web browser compatibility, 306
 authoring, 19
 bandwidth, 21-26
 browser compliance and compatibility, 27
 data rates, 21-25
 document structure, 20-21
 rules, 19-20
 serving your work, 27-30
 tools, 31-35
 basic document, 7-8
 depreciation, 46
 future, 17
 goals, 44
 information Web site, 47
 players and browsers, 10-16
 GRiNS player, 16
 Internet Explorer, 14-16
 QuickTime, 13-14
 RealPlayer, 10-13
 SOJA, 16-17
 profiles, 45
 SMIL 2.0 Basic Language Profile, 45
 XHTML+SMIL 2.0 Language Profile, 45
 reasons to use, 8-10
 specifications, 39
 living in multiple SMIL worlds, 46-47
 SMIL 1.0, 39-43
 SMIL 2.0, 44-46
 SMIL 1.0, 39
 goals, 40
 limitations, 43
 linking abilities, 42
 media layout, 41-42
 switch element, 43
 timing of media, 40-41
 smil element (Structure module), 51
 SMIL Output in Java Applets. *See* SOJA
 SMILBook.com, 48
 SOJA (SMIL Output in Java Applets), 16-17
 soundLevel attribute (Layout module), 79
 sourceLevel attribute (Linking modules), 203
 sourcePlaystate attribute (Linking modules), 204
 specifications, 39
 living in multiple SMIL worlds, 46-47
 SMIL 1.0, 39
 limitations, 43
 linking abilities, 42
 media layout, 41-42
 switch element, 43
 timing of media, 40-41
 SMIL 2.0, 44
 depreciation, 46
 modules, 44-45
 profiles, 45
 speed attribute (Time Manipulations module), 228-229
 SplineAnimation (Animation module), 158-187

- src attribute (Media Object modules), 126
 - srch attribute (RealPix), 252-253
 - srcw attribute (RealPix), 252-253
 - srcx attribute (RealPix), 252-253
 - srcy attribute (RealPix), 252-253
 - stalling, HTML+TIME presentations, 335-337
 - start attribute (RealPix), 253-254
 - startProgress attribute (Transition Effects module), 142
 - streaming architectures, 28
 - pseudostreaming, 28-29
 - true streaming, 29-30
 - Structure module, 44, 49, 345
 - attributes, 54, 345
 - class, 54
 - id, 54
 - title, 55
 - xml:lang, 55
 - xmlns, 55-56
 - conformance profiles, 49-50
 - elements, 50, 345
 - body, 53-54
 - head, 51-52
 - structure of documents (authoring SMIL by hand), 20-21
 - structuring the presentation, HTML+TIME presentations, 311-313
 - subregion positioning, attributes, 81-82
 - backgroundColor, 82
 - bottom, 82
 - fit, 82
 - height, 82
 - left, 82
 - right, 82
 - top, 82
 - width, 82
 - z-index, 82
 - subtype attribute (Transition Effects module), 149
 - SVG (Scalable Vector Graphics), 158, 227
 - switch element (BasicContentControl module), 210-212
 - syncBehavior attribute (Timing and Synchronization module), 108-109
 - syncBehaviorDefault attribute (Timing and Synchronization module), 109
 - Synchronized Multimedia Integration Language.
See SMIL
 - syncMaster attribute (Timing and Synchronization module), 111
 - syncTolerance attribute (Timing and Synchronization module), 109-110
 - syncToleranceDefault attribute (Timing and Synchronization module), 110
 - system-overdub-or-caption attribute (BasicContentControl module), 217
 - systemAudioDesc attribute (BasicContentControl module), 212
 - systemBitrate attribute (BasicContentControl module), 213
 - systemCaptions attribute (BasicContentControl module), 214
 - systemComponent attribute (BasicContentControl module), 214
 - systemCPU attribute (BasicContentControl module), 215
 - systemLanguage attribute (BasicContentControl module), 215-216
 - systemOperatingSystem attribute (BasicContentControl module), 216
 - systemOverdubOrSubtitle attribute (BasicContentControl module), 217
 - systemRequired attribute (BasicContentControl module), 218
 - systemScreenDepth attribute (BasicContentControl module), 218
 - systemScreenSize attribute (BasicContentControl module), 219
- ## T
- tabindex attribute (Linking modules), 204-205
 - tags. *See also* attributes
 - a, 21
 - meta, 21
 - par, 40
 - parallel, 21
 - region, 41
 - seq, 40
 - switch, 21
 - target attribute
 - Linking modules, 205-206
 - RealPix, 254
 - RealText, 268
 - targetElement attribute
 - Animation modules, 163
 - Transition Effects module, 146
 - TCP/IP protocols (Transmission Control Protocol/Internet Protocols), CHECKSUM, 29
 - templates, HTML+TIME presentations, 310-311
 - test attributes (BasicContentControl module), 212
 - text element (Media Object modules), 117
 - textstream element (Media Object modules), 117

- time containers, HTML+TIME presentations, 316-318
- time element (RealText), 259
- Time Manipulations module, 45, 227
 - attributes, 228, 355
 - accelerate, 230-231
 - autoReverse, 232-233
 - decelerate, 231-232
 - speed, 228-229
 - conformance profiles, 227
- timeAction attribute (Timing and Synchronization module), 112-114
- timeContainer attribute (Timing and Synchronization module), 111-112
- timeformat attribute (RealPix), 255
- timing, adding to HTML+TIME presentations, 313-316
- Timing and Synchronization module, 45, 85, 114
 - attributes, 93, 348-349
 - begin, 93-98
 - dur, 98-99
 - end, 93-98
 - endsync, 100
 - fill, 102
 - fillDefault, 103
 - higher, 103-104
 - lower, 104-105
 - max, 99
 - min, 99
 - pauseDisplay, 105-106
 - peers, 106-107
 - repeat, 100-101
 - repeatCount, 101
 - repeatDur, 101
 - restart, 107
 - restartDefault, 108
 - syncBehavior, 108-109
 - syncBehaviorDefault, 109
 - syncMaster, 111
 - syncTolerance, 109-110
 - syncToleranceDefault, 110
 - timeAction, 112-114
 - timeContainer, 111-112
 - conformance profiles, 86
 - elements, 86, 347
 - excl, 90-92
 - par, 88-90
 - priorityClass, 92
 - seq, 86-88
- timing of media, SMIL 1.0, 40-41
- title attribute
 - Media Object modules, 128
 - RealPix, 255
 - Structure module, 55
- tl element (RealText), 260
- to attribute
 - Animation modules, 167
 - Transition Effects module, 143
- tools, authoring, 31
 - Fluition, 32-33
 - GRINS, 32
 - HomeSite, 35
 - Macromedia Flash, 34-35
 - RealSlideshow, 33-34
- top attribute
 - Layout module, 71-72
 - subregion positioning, 82
- topLayout element (Layout module), 69-70
- transIn attribute (Transition Effects module), 146-148
- transition, adding to RealONE Player, 282-285
- Transition Effects module, 45, 133
 - attributes, 138, 350-352
 - begin, 139
 - borderColor, 139
 - borderWidth, 139
 - by, 140
 - calcMode, 140
 - direction, 141
 - dur, 142
 - end, 139
 - endProgress, 142
 - fadeColor, 142
 - from, 143
 - horzRepeat, 143
 - href, 143
 - id, 145
 - mode, 145
 - repeatCount, 145
 - repeatDur, 145
 - startProgress, 142
 - subtype, 149
 - targetElement, 146
 - to, 143
 - transIn, 146-148
 - transOut, 146-148
 - type, 149
 - values, 149-150
 - vertRepeat, 143-144
 - conformance profiles, 136

- elements, 136, 350
 - transition, 137-138
 - transitionFilter, 138
- transition types and subtypes, 151-155
- transition element (Transition Effects module), 137-138
- transition subtypes (Transition Effects module), 151-155
- transition types (Transition Effects module), 151-155
- transitionFilter element (Transition Effects module), 138
- transitions, HTML+TIME presentations, 318
 - continually expanding proprietary set of transitions, 323-324
 - fade-ins and fade-outs, 322
 - opening graphic, 322
- Transmission Control Protocol/Internet Protocols. *See* TCP/IP protocols
- transOut attribute (Transition Effects module), 146-148
- true streaming, 29-30
- tu element (RealText), 260
- tutorial, SMIL application with Internet Explorer, 307-308
- type attribute
 - Animation modules, 165
 - Layout module, 79
 - Media Object modules, 128-129
 - RealPix, 255-256
 - RealText, 268-269
 - Transition Effects module, 149

U

- u element (RealText), 262
- UDP/IP (User Datagram Protocol), 30
- uid attribute (CustomTestAttributes module), 223
- ul element (RealText), 263
- underline_hyperlinks attribute (RealText), 269
- url attribute (RealPix), 256-257
- User Datagram Protocol. *See* UDP/IP

V

- values attribute
 - Animation modules, 169
 - Media Object modules, 129-130
 - Transition Effects module, 149-150
- valuetype attribute (Media Object modules), 130-131
- version attribute (RealText), 269

- vertRepeat attribute (Transition Effects module), 143-144
- video, RealPix, 290-291
- video element (Media Object modules), 117
- viewchange element (RealPix), 242-243
- "Villa Travel"
 - adding transition, 282-285
 - creating the layout, 279-281
 - Flash animation, 281
 - metainformation, 279
 - namespace declaration, 278-279
 - painting the screen with the brush element, 281-282
 - visual plan, 277-278

W

- W3C (World Wide Web Consortium), 39
- Web sites, 357-359
 - authoring resources, 359-361
 - Confluent Technologies, 33
 - Oratrix, 32
 - RealNetworks, 34
 - SMIL 1.0, 361-362
 - SMIL information, 47
- width attribute
 - Layout module, 74-75
 - RealPix, 250
 - RealText, 266
 - subregion positioning, 82
- window element (RealText), 257-258
- wipe effects, RealPix, 287-288
- wipe element (RealPix), 241
- wordwrap attribute (RealText), 269
- World Wide Web Consortium. *See* W3C

X-Y-Z

- x attribute (RealText), 270
- XHTML+SMIL 2.0 Language Profile, 45
- xml:lang attribute
 - Media Object modules, 131
 - Structure module, 55
- xmlns attribute (Structure module), 55-56
- y attribute (RealText), 270
- z-index attribute
 - Layout module, 80-81
 - subregion positioning, 82



SMIL

adding
multimedia
to the web

Synchronized Multimedia Integration Language (SMIL), an XML-based language, allows designers to synchronize and position video, pictures, animations, audio, and text to stream presentations via the Web. Recently, the W3C finalized SMIL 2.0, which offers a more advanced set of controls. Although RealPlayer, QuickTime and Internet Explorer all support SMIL, each supports a slightly different set of commands from SMIL 1.0 and SMIL 2.0.

SMIL: Adding Multimedia to the Web is for designers and developers who want to deliver multimedia content on the Web.

The book begins with an overview of SMIL and the various authoring and delivery platforms. Then, it moves on to cover the SMIL Specification Modules, starting with the basic structure of a SMIL document and building on that knowledge to use SMIL for presentation layout, to create timed animations and links, and to generate transitions and effects. There are concrete code examples for each Module, and a complete Tag reference as well.

Finally, there are real world projects that detail how SMIL is implemented.

SMIL: Adding Multimedia to the Web is a thorough introduction to SMIL, as well as an excellent reference. By the time you finish *SMIL: Adding Multimedia to the Web*, you will be able to create complex multimedia presentations using this exciting new technology.

Tim Kennedy is the originator of Just SMIL (<http://JustSMIL.com>), the first major informational Web site about the SMIL language. His company, Online Delivery, created the original Streaming Media World (<http://www.streamingmediaworld.com>) site for Internet.com. His speaking appearances include informational sessions at Internet World and RealNetworks conferences. An enthusiast for all types of media production, he is currently Chair of the Media Communication and Technology program at Bellevue Community College.

Mary Slowinski is a full-time faculty member at Bellevue Community College (<http://www.bcc.ctc.edu>) instructing in Web design, digital video editing, and streaming technologies. In the summer, she teaches intensive workshops for teachers and other educators to assist them with mastering and incorporating new technologies into their course curriculum. Her other interests include distance speed skating, world music, and viewing vistas.

Category: Multimedia/Graphics

Covers: Synchronized Multimedia Integration Language (SMIL)

User Level: Intermediate

SAMS

www.sampublishing.com

