AdvancED DOM Scripting

Dynamic Web Design Techniques

Jeffrey Sambells
with Aaron Gustafson

friendsof ED™
DESIGNER TO DESIGNER™
an Apress® company

AdvancED DOM Scripting: Dynamic Web Design Techniques

Copyright © 2007 by Jeffrey Sambells, Aaron Gustafson

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13 (pbk): 978-1-59059-856-6

ISBN-10 (pbk): 1-59059-856-3

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit www.apress.com.

The information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

The source code for this book is freely available to readers at www.friendsofed.com in the Downloads section.

Credits

Lead Editor
Chris Mills

Technical Reviewers
Cameron Turner, Victor Sumner

Editorial Board
Steve Anglin, Ewan Buckingham, Gary Cornell, Jonathan Gennick, Jason Gilmore, Jonathan Hassell, Chris Mills, Matthew Moodie, Jeffrey Pepper, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh

Senior Project Manager
Kylie Johnston

Copy Edit Manager
Nicole Flores

Copy Editor
Heather Lang

Assistant Production Director
Kari Brooks-Copony

Senior Production Editor
Laura Cheu

Compositor
Dina Quan

Artist
Kinetic Publishing Services, LLC

Proofreader
Liz Welch

Indexer
Broccoli Information Management

Cover Image Designer
Bruce Tang

Interior and Cover Designer
Kurt Krames

Manufacturing Director
Tom Debolski

CONTENTS

About the Authors . . . . **xv**

About the Technical Reviewers . . . . **xvi**

About the Cover Image Designer . . . . **xvii**

Acknowledgments . . . . **xviii**

Introduction . . . . **xix**

PART ONE **DOM SCRIPTING IN DETAIL**

Chapter 1 **Do It Right with Best Practices** **3**

- Unobtrusive and progressive enhancement 4
- Putting JavaScript to work 5
  - Separating your behavior from your structure. 6
    - How to include JavaScript the right way 6
    - That javascript: prefix 8
  - Don't version check! 14
    - Use capability detection 14
    - When browser version sniffing is OK 16
  - Degrade gracefully for guaranteed accessibility 16
    - Don't require JavaScript for content—period. 16
  - Plan for reuse with namespaces 17
  - Simplify things with reusable objects 19
    - Beginning the ADS library 20
    - The ADS.isCompatible() method 21
    - The ADS.$() method 22
    - The ADS.addEvent() and ADS.removeEvent() methods 24
    - The ADS.getElementsByClassName() method 26
    - The ADS.toggleDisplay() method 28
    - The ADS.insertAfter() method 28
    - The ADS.removeChildren() and ADS.prependChild() methods 29
  - Get your hands dirty 30

Common gotchas in the JavaScript syntax . . . 30
Case sensitivity. . . . 30
Single vs. double quotes . . . 31
Breaking lines . . . 31
Optional semicolons and parentheses . . . 32
Overloading (not really) . . . 33
Anonymous functions. . . . 33
Scope resolution and closures . . . 34
Iterating over objects . . . 39
Referencing vs. calling a function (missing parentheses) . . . 40
A practical example: WYSIWYG JavaScript rollover redux . . . 40
Summary . . . 48

Chapter 2 **Creating Your Own Reusable Objects** . . . **51**

What's in an object?. . . . 52
Inheritance . . . 53
Understanding object members . . . 53
Everything's in the window object. . . . 56
Making it all possible with scope and closure . . . 60
Creating your own objects . . . 60
One becomes many: creating the constructor . . . 61
Adding static methods . . . 63
Adding public methods to the prototype. . . . 64
Controlling access with private and privileged members . . . 65
Do public, private, privileged, and static really matter? . . . 67
The object literal . . . 68
What is this? . . . 71
Redefining your context with call() and apply() . . . 73
try { }, catch { }, and exceptions . . . 76
A practical example: your own debugging log . . . 78
Why use a JavaScript logging object? . . . 78
The myLogger() object . . . 78
The myLogger.createWindow() method . . . 80
The myLogger.writeRaw() method. . . . 82
The myLogger.write() and myLogger.header() methods. . . . 85
Summary . . . 87

Chapter 3 **Understanding the DOM2 Core and DOM2 HTML** . . . **89**

The DOM, not JavaScript, is your document . . . 90
Objects and interfaces . . . 90
Levels of the DOM . . . 91
DOM Level 0. . . . 91
DOM Level 1. . . . 91
Level 2 . . . 91
Level 3 . . . 92
Which level is correct for you?. . . . 93

Creating a sample document 96
Creating the DOM file 96
Choosing a browser 98
The DOM Core 100
The importance of inheritance in the DOM 102
The Core Node object 103
Node names, values, and types 103
Node parents, children, and siblings 108
Node attributes 111
The node ownerDocument property 113
Checking for children and attributes 114
Manipulating your DOM node tree 115
Duplicating and moving a node 117
The Core Element object 119
Manipulating Element attributes 119
Locating Element objects within Element objects 120
The Core Document object 120
The document.documentElement property 121
Creating nodes with document methods 121
Locating Elements with Document methods 122
Traversing and iterating the DOM tree 122
DOM HTML 125
The DOM2 HTML HTMLDocument object 126
The HTML HTMLElement object 127
A practical example: converting hand-coded HTML to DOM code 127
The DOM generation tool HTML file 128
Testing with an example HTML fragment 130
Adding to the ADS library 131
The generateDOM object framework 133
The encode() method 133
The checkForVariable() method 134
The generate() method 134
The processNode() and processAttribute() methods 136
Summary 146

Chapter 4 **Responding to User Actions and Events** **149**

DOM2 Events 150
Types of events 151
Object events 151
The load and unload events 151
The abort and error events 152
The resize event 153
The scroll event 153
Mouse movement events 153
Mouse click events 156
Keyboard events 159

Form-related events 159
Form submit and reset events 159
Blur and focus events 161
The change event 163
W3C DOM-specific events 165
Custom events 166
Controlling event flow and registering event listeners 166
Event flow 167
Order of events 171
Two phases and three models 173
Popping the bubble 175
Cancelling the default action 176
Registering events 178
Inline registration model 178
The ADS.addEvent() method revisited 178
The traditional event model 179
Microsoft-specific event model 181
W3C DOM2 Events model 181
The problem with the load event 183
Accessing the event object from the event listener 186
Syntactical shortcuts 187
The ADS.getEventObject() method 187
Cross-browser event properties and methods 188
The DOM2 Events object 189
The DOM2 MouseEvent object 190
Browser incompatibilities galore 192
Accessing keyboard commands 197
Summary 201

Chapter 5 **Dynamically Modifying Style and Cascading Style Sheets 203**

The W3C DOM2 Style specification 203
CSSStyleSheet objects 204
CSSStyleRule objects. 204
CSSStyleDeclaration 205
A lack of support 206
When DOM scripting and style collide 206
Modifying markup for style 207
Removing the extra markup 210
Keeping style out of your DOM script 213
The style property 213
Switching styles based on a className 217
Using common classes with className switching 217
Drawbacks of using className switching 220
Why not use setAttribute for class names? 220

Switching the style sheet 220
Using alternative style sheets 221
Switching the body className 225
Dynamically loading and removing style sheets 228
Modifying CSS rules 229
AdvancED image replacement revisited 234
Accessing the computed style 237
The Microsoft filter property 239
Practical example: a simple transition effect 244
Summary 247

Chapter 6 **Case Study: A Photo Cropping and Resizing Tool** **249**

The test files 250
The editor objects 254
Invoking the imageEditor tool 259
The imageEditor load event. 260
Creating the editor markup and objects. 262
Adding the event listeners to the editor objects 270
Resizing the image 272
Cropping the Image 276
The incomplete image editor 280
Summary 281

PART TWO **COMMUNICATING OUTSIDE THE BROWSER**

Chapter 7 **Adding Ajax to the Mix** **285**

Merging technology 286
Semantic XHTML and the DOM 286
JavaScript and the XMLHttpRequest object 286
Making a new request. 287
Acting on the response 288
Identifying Ajax requests on the server 291
Beyond GET and POST 294
XML 295
Plain text 296
HTML 296
JavaScript code. 297
JSON 298
A reusable object 299
Is Ajax right for you? 304
Why Ajax may break your site and how to fix it 305
JavaScript required for content. 305
Bypassing cross-site restrictions with <script> tags. 306

Back buttons and bookmarks . . . 313
A not so simple fix . . . 315
Browser sniffing for product features . . . 316
Tracking location changes . . . 316
A race to finish the request . . . 324
Latency picks the winner . . . 327
Dealing with asynchronous requests . . . 329
Increased resources . . . 334
Problems solved? . . . 334
Practical example: an Ajax-enhanced photo album . . . 334
Ajaxify the photo browser . . . 338
Summary . . . 343

Chapter 8 **Case Study: Enabling Asynchronous File Uploads with Progress Indicators . . . 345**

A little life in the loading message . . . 347
Processing uploads on the server . . . 349
The magic word . . . 351
The starting point . . . 351
Putting it all together: an upload progress indicator . . . 352
The addProgressBar() framework . . . 355
The load event . . . 357
The addProgressBar() object . . . 358
Modifying the file inputs . . . 358
Redirecting the form . . . 361
And the magic word is . . . 362
The progress bar . . . 363
Tracking progress . . . 365
Summary . . . 372

PART THREE **SOME GREAT SOURCE**

Chapter 9 **Using Libraries to Increase Productivity . . . 375**

Choosing the library that's right for you . . . 376
The libraries . . . 377
DOMAssistant . . . 378
jQuery . . . 378
Mochikit . . . 378
Prototype . . . 379
Yahoo User Interface library . . . 379

Enhancing the DOM 380
Chaining syntax 380
Advanced selection with expressions 382
jQuery with XPath 385
Filtering with a callback 387
Manipulating the DOM document 389
Using DOMAssistant to create elements 389
Using jQuery to move nodes 389
Using MochiKit to create elements 390
Using Prototype to clean up your document 390
Using YUI to check for intersecting elements. 390
Iterating over results 391
Handling events 391
Registering events 392
The DOMAssistant way 392
The jQuery way 393
Custom events 395
Accessing and manipulating style. 396
Communication 397
Prototype Ajax object 397
jQuery keeps Ajax simple 400
Summary 402

Chapter 10 **Adding Effects to Enhance User Experience**. **405**

Do it yourself 406
Show me the content!. 407
Providing feedback 411
The Yellow Fade Technique 411
Avoiding shifting content 412
A few visual effects libraries 414
Moo.fx 414
Script.aculo.us 414
Some visual bling. 415
Mootastic CSS property modification 415
One property at a time 416
A mix of properties all at once. 417
Reusing the effect 417
Multiple effects on multiple objects 418
Sliding with Moo.fx 418
Form feedback made pretty 420
Visual effects with Script.aculo.us 427
Parallel effects 429
Realistic motion using Moo.fx 430
Customer form revisited 434
Rounding corners 435
The rest of the libraries 437

Behavioral enhancements 437
Drag and drop with Script.aculo.us 438
Drag anywhere 438
Dropping on a target: the droppable 439
Building a drag-and-drop shopping cart with Script.aculo.us 440
Interacting with draggables through an observer 448
More drag and drop fun 451
Summary 452

Chapter 11 **Mashups Galore! Using APIs to Add Maps, Searching, and Much More 455**

API keys 457
Client-side APIs: some JavaScript required 457
Maps put mashups on the map 458
Retrieving latitude and longitude 464
Maintaining accessibility using microformats 465
Ajax search requests 470
Search results for your site only 476
Related links 477
Mashing Search with Maps 481
Server-side APIs: some proxy required 484
An integrated to-do list with Basecamp 488
Your Basecamp account information 489
Building the Basecamp proxy 491
The Basecamp DOM script 495
Buddy icons with Flickr 498
The Flickr API key 499
Building the Flickr proxy 499
The DOM script 502
Summary 504

Chapter 12 **Case Study: Style Your select with the DOM 507**

That classic feeling 508
Building a better select 509
Strategy? We don’t need no stinkin’ strategy 510
The files 511
The FauxSelect objects 512
Getting the faux select going 514
Locating the select elements 515
A little housekeeping 517
Building the DOM elements 518
Creating a faux value 520
Creating faux options 521
Generating life and other memorable events 522
Opening, closing, and clicking the select 522
Selecting a faux option 525

Bling-bling for da form t'ing 527
Behavioral modifications 538
Closing the faux select 538
Z index to the rescue! 542
Keyboard controls and other niceties 543
Selecting options 543
Maintaining focus 544
Closing the faux select 546
Is select too big for its britches? 549
Knock, knock . . . housekeeping! 551
Further adventures in select replacement 552
Summary 553

Index 555