

Index

COPYRIGHTED MATERIAL

Index

NUMBERS

“The 20-Minute Wiki” Web address, 29
 37Signals interview (Fried and Henemeier
 Hansson), 31–34

SYMBOLS

“ (double quotes), using in Dojo, 260, 526
 & separator, using with HTTP request for
 mashup, 75
 : (colon), using in Dojo, 526
 ? separator, using with HTTP request for
 mashup, 74
 [] (square brackets), meaning in Dojo, 260, 527
 {} (curly braces), using in Dojo, 260, 526
 < (opening bracket), using in XML tags, 37
 > (closing bracket), using in XML tags, 37

A

a element in XHTML, meaning of, 47
 abort() method of XMLHttpRequest object,
 description of, 73
 accept Droppable option, description of, 331
Action Controller, role in Ruby on Rails MVC
 framework, 179
 action element, adding to end of Widget
 specification, 475–477
Action View, role in Ruby on Rails MVC framework,
 179

Active Record, role in Ruby on Rails MVC
 framework, 179
ActiveRecord Migration, schema definitions as,
 350
 acts_as_taggable plug-in, installing for
 tagging, 432–434
 acts_as_taggable.rb, 444
 add Action, using with Timeline controller, 364
 add_concept function for API, 462
 add_image_concept method, using with Stax,
 460–461, 464
 AddressLocale.java for server-side form
 usability, 300–301
 addSlideToTimeline function for client-only
 ideaStax Editor, 346
 afterFinish argument, using with
 script.aculo.us effects, 372
 afterUpdate argument, using with
 script.aculo.us effects, 372
agile languages
 in RIAs (interview with 37Signals), 32
 and scalability (interview with 37Signals), 33
 studies about, 26
AJAX (Asynchronous JavaScript and XML). See
 also Web applications
 advantages over traditional approach, 125
 components of, 125
 conceptualizing, 8
 using with Web applications, 123–125
AJAX action, occurrence in RJS construct for
 partial renderer, 242

AJAX approach to Web applications, 124

AJAX edition of Dojo, downloading, 258

AJAX effects, advantages in forms, 297–298

AJAX support

in Dojo JavaScript library, 193

in JavaScript libraries, 190

in MochiKit JavaScript library, 196

in script.aculo.us JavaScript library, 199

in YUI (Yahoo! UI) JavaScript library, 202

AJAX word processor, using, 11

Ajaxos Web address, 20

alert.js code sample, 56

alert(String) function, using in logging, 158

altered minds, observations from, 23–24

Angela's Ristorante wait-staff portal, 62–67

animations

in Dojo JavaScript library, 192

in JavaScript libraries, 190

in MochiKit JavaScript library, 196

in script.aculo.us JavaScript library, 199

in YUI (Yahoo! UI) JavaScript library, 201

Apache Web address, 71

Apache Web server, using in tagging infrastructure, 397–398

API authentication process, 457–458, 458

API calls, form of, 455

API for Stax

documentation sample for, 459–462

writing documentation for, 459

API keys, using, 456

APIs (application programming interfaces)

adding API method for, 463–465

adding authentication to, 465

casual versus formal solutions for, 455

controlling user access to, 455–456

creating new image concept for, 462

defining interface for, 458–459

exposing, 15

home-grown solution for, 454, 459

planning, 455–456

saving uploaded image file for, 462–463

using metered APIs, 455–458

Web service solution for, 454–455

Appear effect in script.aculo.us, argument for and description of, 370

.appendChild(element) method of DOM element, use of, 60

Apple Dashboard Widgets, description of, 468

application development, changing approach toward, 27–28

application flow control, comparing in Java and Ruby on Rails, 180

application programming interfaces (APIs)

adding API method for, 463–465

adding authentication to, 465

casual versus formal solutions for, 455

controlling user access to, 455–456

creating new image concept for, 462

defining interface for, 458–459

exposing, 15

home-grown solution for, 454, 459

planning, 455–456

saving uploaded image file for, 462–463

using metered APIs, 455–458

Web service solution for, 454–455

applications

considering as betas, 27

relationship to operating systems, 12

Applications Are Always Services pattern, 16

asynchronous form validation, explanation of, 251

Asynchronous JavaScript and XML (AJAX)

advantages over traditional approach, 125

components of, 125

conceptualizing, 8

using with Web applications, 123–125

asynchronous requests

creating for Hello World example, 128

relationship to JavaScript libraries, 186–187, 186–188

asynchronous validation, adding for username, 289–291

at_rot_target event in LSL, trigger associated with, 505

at_target event in LSL, trigger associated with, 505

attach event in LSL, trigger associated with, 505

attribute information, extracting from RIAPage requests, 83–84

attributes

defining for server-side form validation, 279–281

reference for Dojo validation Widgets, 265

using in XML, 38–39

using with Dojo validation Widgets, 261

using with Widgets, 473

`.attributes` **property of DOM element, use of, 60**
Au, Wagner James (Second Life interview), 519–522
authentication, adding to APIs, 465
`autoComplete` **property of ComboBox Widget, default value and meaning of, 317**
avatar exercising quote object and server, 518

B

`before_filter` **command, using with API, 465**
`beforeStart` **argument, using with `script.aculo.us` effects, 372**
`beforeUpdate` **argument, using with `script.aculo.us` effects, 372**
betas, considering applications as, 27
BIG DESIGN waterfall method, 29
`BlindDown` **effect in `script.aculo.us`, argument for and description of, 370**
`BlindUp` **effect in `script.aculo.us`, argument for and description of, 370**
blot, mitigating, 11
`blockquote` **element in XHTML, meaning of, 45**
`body` **element in XHTML, meaning of, 43**
The Browser Creates User Experience pattern, 16
browser-centric nature of RIAs, 21
browsers
 handling, 379–380
 instantiation of, 92–93
 logging added to, 157
bugs in applications, considering, 27
Builder tool in `script.aculo.us`
 creating slides dynamically with, 336–339

C

callback function
 defining for client-side form usability, 306–307
 defining for form validation, 270
 defining for Hello World, 130–131
 defining for server-side username validation, 291
 extending for client-side form validation, 283–284
 specifying for client side of combo box, 319–320
“callback” parameter, using with Yahoo and Google Traffic mashup, 103–104
car make combo box example
 adding event handler for client side, 318
 defining controller for, 312–314

 defining model for, 311–312
 description of, 309–311
 implementing client side for, 314–321
 implementing server side for, 311–314
 specifying callback function for client side, 319–320
 updating `onLoad` function for client side, 321
 updating select element for client side, 318
 using Dojo combo boxes with, 314–317

Card in `ideaStax`, 226

Card thumbnail in `ideaStax`, 227

Card with Concept in `ideaStax`, 228

`card_controller` **for tagging, 440**

cardlist renderer, implementing for tagging, 441–443

`_cardlist.rhtml`, **441**

`card.rb` **for `ideaStax`, 234**

Cards, displaying in preview area of `Stax`, 425

Cards in `ideaStax`

 creating, 226–229

 description of, 222–223

 validation requirements for, 235

 verifying, 235–236

Cards panel, displaying Cards in, 424

`cardship.rb` **for `ideaStax`, 235**

cardships model, explanation of, 234

CASCADING, appearance in red, 53

cascading style sheets (CSS)

 applying to XHTML elements, 53

 attaching to Web pages, 50–51

 benefits of, 48–49

 description of, 20, 36

 relationship to DOM elements, 327

 specifying styles with, 51

 styling based on element class, 52

 styling based on element ID, 52

 styling based on element’s ancestry, 53

`caseSensitive` **property of ComboBox Widget, default value and meaning of, 317**

changed event in LSL, trigger associated with, 505

Chat Catcher in Second Life, 514

`Chat_Catcher` **LSL for Second Life, 511–512**

`check_api_key` **function, executing, 465**

CherryPy framework, relationship to TurboGears, 181

`.childNodes` **property of DOM element, use of, 60**

client application, enhancing for RIAs, 142–145

client interactions with Widgets, examples of, 482

client side

- adding event handler for combo box, 318
- specifying callback function for combo box, 319–320
- updating onload function for combo box, 321
- updating select element for combo box, 318
- client-only ideaStax Editor. See also ideaStax Editor; server-based ideaStax Editor**
 - creating droppable Editor pane for, 342–344
 - creating slides with, 336–339
 - description of, 336
 - implementing `Sortable` behavior for, 345–346
 - implementing sortable Timeline for, 344–346
 - populating photo library with draggable slides, 339–342
- client-side form usability. See also server-side form usability**
 - defining callback function for, 306–307
 - defining event handler function for, 305–306
 - implementing, 305–308
 - invoking event handler on zip code change, 308
- client-side form validation**
 - explanation of, 251
 - implementing by enhancing view, 282–284
- client-side JavaScript, benefit of, 296**
- client-side query generator in Web 2.0, 515–516**
- client-side validation with Dojo. See also Dojo validation Widgets**
 - functions in, 259–260
 - installing Dojo Toolkit for, 258–259
- close PreviewMode function, amending for ideaStax Previewer modal window, 385**
- closing bracket (>), using in XML tags, 37**
- CocoaMySQL database tool, using with tagging, 434–437**
- code samples. See also Listings**
 - asynchronous request related to JavaScript libraries, 186–187
 - `lookupCities` method for combo box, 311–312
 - style added to Web page, 6
 - `submitCallback` function, 283
 - Web page with DOM elements, tag pairs, and JPEG image, 5
 - Widget specification, 471
- collaboration via Web, requirements for, 26**
- collections, representing with LSL, 504**
- collision event in LSL, trigger associated with, 505**

- collision_end event in LSL, trigger associated with, 505**
- collision_start event in LSL, trigger associated with, 505**
- colon (:), using in Dojo, 526**
- combo box example, 309–310, 320. See also form usability**
 - adding event handler for client side, 318
 - defining controller for, 312–314
 - defining model for, 311–312
 - description of, 309–311
 - implementing client side for, 314–321
 - implementing server side for, 311–314
 - specifying callback function for client side, 319–320
 - updating onload function for client side, 321
 - updating select element for client side, 318
 - using Dojo combo boxes with, 314–317
- combo boxes. See also Dojo combo boxes**
 - properties of, 316–317
 - specifying options in, 315–316
- ComboBox Widget, properties of, 316–317**
- CometD, interview with Alex Russell, 323–324**
- command and DAO in Model 2 Java architecture, 173**
- commands, managing for Stax API Widget, 489–490**
- community tools, 178**
- compiled-to-JavaScript approach. See GWT (Google Web Toolkit); RJS (Ruby JavaScript)**
- Concept class, using with tagging, 435**
- Concept dragged in ideaStax, 227**
- concept for Stax API Widget, uploading to Stax server, 488–490**
- Concept in ideaStax, 225**
 - `concept` instance, array of tags for, 436
 - Concept model, description of, 234**
 - `concept_controller.rb` in ideaStax, 239
 - `concept.rb`
 - for ideaStax, 234
 - for tagging, 434
- Concepts in ideaStax**
 - adding, 226
 - creating, 224–225
 - description of, 222–223
 - verifying, 235–236
- console**
 - inclusion in Web application frameworks, 178
 - verifying Concepts, Cards, and Stacks with, 235
- console command, validating tagging with, 434–437**

- console output, logging as, 156**
 - constraint `script.aculo.us` **Draggable** option, **description of, 330**
 - containment diagram for Angela's Ristorante, 62**
 - containment
 - Droppable option, 331
 - sortable list option, **description of, 332**
 - content, relationship to mediating applications, 12.**
See also Web content
 - content editors, example of, 21**
 - content **property, using with `dojo.io.bind` function, 290**
 - control **event in LSL, trigger associated with, 505**
 - controller filtering, comparing in Java and Ruby on Rails, 180**
 - controller framework, 177**
 - controller in MVC**
 - controversy about, 169
 - for Model 2 Java architecture, 171
 - controller inheritance, 177**
 - controller responsibilities in MVC pattern, 167**
 - controllers**
 - creating for server-backed ideaStax Editor, 360–364
 - defining for combo box example, 312–314
 - defining for form usability, 303–305
 - defining for server-side form validation, 276–277
 - enhancing for server-side username validation, 291–292
 - implementing in ideaStax, 239
 - implementing in tagging infrastructure, 437–441
 - in Ruby on Rails MVC framework, 360
 - controllers **subfolder in Rails, description of, 244, 445**
 - `copySlide()` **function, using with editor pane, 342**
 - counter-manifesto patterns. See also patterns**
 - 1: Expose the Remote API, 15
 - 2: Use a Common Data Format, 15
 - 3: The Browser Creates the User Experience, 16
 - 4: Applications Are Always Services, 16
 - 5: Enable User Agnosticism, 16
 - 6: The Network Is the Computer, 16–17
 - 7: Web 2.0 Is a Cultural Phenomenon, 17
 - country parameter, providing for `helloResponsev2`, 138**
 - `create_cardship.rb` **for ideaStax, 236**
 - `create_concepts.rb` **for Ruby on Rails, 233**
 - `create_taggings.rb`, **433**
 - `create_tags.rb`, **433**
 - `.createAttribute()` **method, use of, 60**
 - `.createElement()` **method, use of, 59**
 - `.createEvent()` **method, use of, 60**
 - `.createRange()` **method, use of, 60**
 - `create.rhtml` **for ideaStax, 240**
 - `.createTextNode()` **method, use of, 59**
 - CSS (cascading style sheets)**
 - applying to XHTML elements, 53
 - attaching to Web pages, 50–51
 - benefits of, 48–49
 - description of, 20, 36
 - relationship to DOM elements, 327
 - specifying styles with, 51
 - styling based on element class, 52
 - styling based on element ID, 52
 - styling based on element's ancestry, 53
 - CSS applied to HTML for Angela's Ristorante, 66**
 - CSS development tips, 54**
 - CSS files**
 - advisory about including HTML tags in, 80
 - components of, 51
 - CSS instructions, placing in header of XHTML documents, 50**
 - CSS properties for drag and drop operations, 328**
 - CSS styles, defining by element ID and element class, 66**
 - CSV, defining Ruby fixtures with, 353**
 - `curl`, **invoking for Stax API Widget, 488–489**
 - curly braces ({}), using in Dojo, 260, 526**
 - currency value, testing in Dojo, 527–528, 531**
 - `CurrencyTextbox` **Dojo validation Widget, function and test related to, 264**
- ## D
- Damn Small Linux Web address, 21**
 - DAOs (Data Access Objects), using in Java Health History design, 174**
 - data, accessing across domains, 108**
 - data access details, comparing in Java and Ruby on Rails, 180**
 - Data Access Objects (DAOs), using in Java Health History design, 174**
 - data definition details, comparing in Java and Ruby on Rails, 180**
 - data format, using common format, 15**

data mining, 400–401

data model, setting up for photos and tags, 403–408

data storage details, comparing in Java and Ruby on Rails, 180

data tables, setting up for photos and tags, 404–408

database, setting up for server-side form usability, 299–300

database interaction abstraction, 177

database lookups, performing in Tag Cloud application, 418–420

database schema, defining for server-backed ideaStax Editor, 350–351

database to model object integration

inclusion in Web application frameworks, 177

dataProvider property of ComboBox Widget, default value and meaning of, 317

dataserver event in LSL, trigger associated with, 505

date formats, testing in Dojo, 528–529

DateTextbox Dojo validation Widget, function and test related to, 264

dbHandler.py for Flickr photo interface, 413–414

DBLookup.py for Tag Cloud application, 418–420

Debug Console Widget, adding for form validation, 260–262

debugging

availability in GWT, 207

in Dojo JavaScript library, 192

in JavaScript libraries, 190

in MochiKit JavaScript library, 195

in script.aculo.us JavaScript library, 198

using CSS for, 50

in YUI (Yahoo! UI) JavaScript library, 201

debugging tools

FireBug, 154–155

Firefox, 150–151

JavaScript console, 152

markup validation, 155–156

Mozilla DOM Inspector, 151–152

Venkman, 152–153

design, mitigating bad design, 11

desktop, leaving behind, 20

desktop applications

versus RIA development styles, 18

significance of, 13

desktop programming, process focused nature of, 35

developer key for API, 456

Digit Dojo validation Widget attribute, default value and meaning of, 265

direction argument, using with script.aculo.us effects, 372

div element in XHTML

meaning of, 43

use of, 44

!DOCTYPE element in XHTML, meaning of, 48

document object model (DOM)

applying style sheet to, 5

inclusion in AJAX, 125

DocumentRoot in MochiKit, 399

doGet method, using with

HelloResponseServletv2.java, 141

Dojo combo boxes. See also combo boxes

defining, 317

using with forms, 314–316

Dojo functions, using in form validation, 269–271

Dojo JavaScript library

advantages and disadvantages of, 194

capabilities of, 191–193

description of, 188–189

including in Web applications, 259

Internet validation functions and flags, 529–530

locale-specific validation functions in, 531–532

quick reference for, 204

using client-side validation functions in, 259–260

validation functions and flags in, 526–529

Dojo logger console, 192

Dojo Toolkit

installing for client-side form validation, 258–259

interview with Alex Russell, 322–324

Dojo validation Widgets. See also client-side validation with Dojo

attribute reference for, 265

Debug Console Widget, 260–262

field validation Widgets, 265–268

reference for, 264

Username Field Validation Widget, 262–263

Dojo Widgets used for Web application, 193

dojo.io.bind function, using, 187, 290

DOM (document object model). See also Mozilla DOM Inspector debugger

applying style sheet to, 5

inclusion in AJAX, 125

DOM elements

- methods of, 60–61
- multiple classes of, 52
- properties of, 60
- relationship to CSS, 327

DOM Inspector's interface, 151**DOM regions, replacing with RJS, 219****DOM structures, creating with Builder tool, 337****DOM support**

- in Dojo JavaScript library, 193
- in JavaScript libraries, 190
- in MochiKit JavaScript library, 196
- in script.aculo.us JavaScript library, 200
- in YUI (Yahoo! UI) JavaScript library, 202

DOM tree

- accessing from JavaScript, 59–60
- creating elements for insertion into, 59–60
- locating elements in, 59
- relationship to XML, 39–42
- for RIAPage requests, 82–83
- showing and hiding elements in, 368

domains, accessing data across, 108**doQuery function, using with RIAPage requests, 83–84****double quotes (“), using in Dojo, 260, 526****do-while flow control keyword in LSL, description of, 509****drag and drop components**

- in Dojo JavaScript library, 192
- in JavaScript libraries, 190
- in MochiKit JavaScript library, 195
- in script.aculo.us JavaScript library, 199
- in YUI (Yahoo! UI) JavaScript library, 201

drag and drop operations. See also ideaStax Editor

- adding to Stax API Widget, 486–487
- CSS properties for, 327–328
- planning for, 335
- relationship to sortable lists, 332

drag appearance, considering for Draggable objects, 329**drag behavior, utilizing in RIAs, 329****drag constraints, considering for Draggable objects, 329****Draggable class**

- defining, 329–330
- description of, 328

effect bindings for, 374–375

requirements for, 328–329

draggable slides, populating photo library with, 339–342**drop-down lists, including in combo boxes, 315****dropOnEmpty sortable list option, description of, 332****DropOut effect in script.aculo.us, argument for and description of, 370****Droppable behavior, implementing for client-only ideaStax Editor, 344–346****Droppable class**

callbacks on, 375–376

description of, 330

requirements for, 330–331

droppable Editor pane, creating for client-only ideaStax Editor, 342–344**Droppable elements, defining, 331****duration argument, using with script.aculo.us effects, 371****dynamic scripting, use of, 108. See also scripting code****DynamicScripting.html, 104–105****E****Editor controller, building for server-backed ideaStax Editor, 361–362****Editor in ideaStax Editor, description of, 333, 335****Editor pane, creating for client-only ideaStax Editor, 342–344****Editor panel**

in ideaStax, 344

in Stax, 426–427

Editor view, constructing for server-backed ideaStax Editor, 359–360**Effect.Parallel class, combining effects with, 377****effects**

adding with Sortable objects, 377

binding to script.aculo.us actions, 374–377

in Dojo JavaScript library, 192

in JavaScript libraries, 190

in MochiKit JavaScript library, 196

in script.aculo.us JavaScript library, 199, 369–372

in YUI (Yahoo! UI) JavaScript library, 201

effects-based notifications example, 372–374

- element ancestry, basing CSS styles on, 53**
- element class, basing CSS styles on, 52, 66**
- element contents, inspecting and setting in JavaScript, 60–61**
- element ID, basing CSS styles on, 52, 66**
- element style manipulation**
 - in Dojo JavaScript library, 192
 - in JavaScript libraries, 190
 - in MochiKit JavaScript library, 196
 - in script.aculo.us JavaScript library, 199
 - in YUI (Yahoo! UI) JavaScript library, 201
- elements**
 - creating for inserting into DOM tree, 59–60
 - hiding and showing, 368–369
 - locating in DOM tree, 59
 - order in XHTML, 44
 - substituting with attributes in Widgets, 473
- elements in XML, modeling items as, 38**
- em element in XHTML, meaning of, 45**
- e-mail address formatting conventions, testing in Dojo, 530**
- email event in LSL, trigger associated with, 505**
- e-mail versus instant messaging, 9**
- EmailListTextbox Dojo validation Widget, function and test related to, 264**
- EmailTextbox Dojo validation Widget, function and test related to, 264**
- endeffect property, using with Draggable class, 375**
- endeffect script.aculo.us Draggable option, description of, 330**
- eval function in JSON, advisory about, 92**
- event handler, adding for client side of combo box, 318**
- event handler function**
 - defining for client-side form usability, 305–306
 - defining for Hello World example, 128–130
- event handler, invoking on zip code change in client-side forms, 308**
- event-handling enhancements**
 - in Dojo JavaScript library, 193
 - in JavaScript libraries, 190
 - in MochiKit JavaScript library, 197
 - in script.aculo.us JavaScript library, 200
 - in YUI (Yahoo! UI) JavaScript library, 202
- eXtensible HyperText Markup Language (XHTML)**
 - avoiding HTML elements in, 47
 - describing text type in, 45–46

- description of, 36
- document setup in, 47–48
- document structure of, 43–44
- features of, 42
- versus HTML, 47–48
- in-text objects in, 46–47
- order of elements in, 44
- origins of, 43
- in RIAs, 49
- text structure of, 44–45

- eXtensible Markup Language (XML)**
 - attributes in, 38–39
 - elements and attributes in, 37–38
 - description of, 35
 - versus JSON, 96
 - purpose of, 36
 - relationship to DOM tree, 39–42
 - using JSON as alternative to, 86

F

- Fade effect in script.aculo.us, argument for and description of, 370**
- fadeTime property of ComboBox Widget, default value and meaning of, 316**
- :fail option, providing for remote calls, 376**
- FCkEditor Web address, 21**
- field validation Widgets in Dojo, adding, 265–268**
- fields**
 - adding validation to, 284–288
 - validating in server-side forms, 280–281
- Figures**
 - AJAX approach to Web applications, 124
 - API authentication process, 458
 - avatar exercising quote object and server, 518
 - Card in ideaStax, 226
 - Card thumbnail in ideaStax, 227
 - Card with Concept in ideaStax, 228
 - Chat Catcher in Second Life, 514
 - CocoaMySQL database tool used with tagging, 434
 - combo box example, 320
 - combo box example of form usability, 309–310
 - command and DAO in Model 2 Java architecture, 173
 - Concept dragged in ideaStax, 227
 - Concept in ideaStax, 225
 - containment diagram for Angela’s Ristorante, 62
 - CSS applied to HTML for Angela’s Ristorante, 66

- CSS properties for drag and drop operations, 328
- DocumentRoot in MochiKit, 399
- Dojo logger console, 192
- Dojo Widgets used for Web application, 193
- DOM Inspector's interface, 151
- Editor panel in ideaStax, 344
- Editor panel in Stax, 427
- FireBug in Layout Inspector mode, 154
- FireBug plug-in for Firefox, 394
- form before username specification, 263
- form submissions, 272
- front controller execution in Java, 172
- geocoding example, 98
- Google as invisible application, 10
- Google's Gmail interface, 372–373
- Google's Map API, 101
- GWT application running, 211
- GWT hosted server and browser, 212
- GWT project directory tree, 209
- Health History design in Java, 174
- Hello World example, 145
- Hello World RIA Widget packed with Widget Converter, 475
- Hello World Widget, 480
- Helloworld page, 127
- HTML document without CSS file, 64
- ideaStax application partial render, 242
- ideaStax Editor, 333
- ideaStax Editor with slides rearranged, 346
- ideaStax hierarchy, 222
- ideaStax interface, 223
- IdeaStax Library after loading photos, 342
- IdeaStax Library before loading photos, 341
- ideaStax Previewer modal window layout calculations, 380
- JavaScript console, 153
- JavaScript console displaying error, 85
- JSON deconstruction, 88
- logging messages, 160
- LSL multi-state script in action, 508
- LSL New Script template, 510
- Magic Cap interface, 498–499
- mashup in JavaScript, 81
- MashupFeed selection, 70
- MochiKit inline logging pane, 196
- MochiKit logger, 157
- modal preview, 386
- MVC (Model-View-Controller) pattern, 165
- MVC pattern high-level class diagram, 166
- MVC pattern in Java's Model 2 architecture, 170
- MVC pattern variation, 168
- notification in ideaStax, 374
- Open Widget dialog box, 474
- Photos table snapshot, 355
- Python "Easter Egg," 24
- Rails application flow control model, 443
- rake command output, 354
- rendered form for tagging, 440
- RIA objects, 347
- RIA solutions, 19
- RIAPage for Google Map API, 107
- RJS development, 217–218
- Ruby on Rails flow control model, 243
- Ruby on Rails MVC framework Health History design, 180
- script.aculo.us JavaScript library autocompleter, 199
- Second Life environment, 501
- server-side username validation, 292
- Stack, Card, and Concept in ideaStax, 229
- Stack in ideaStax, 229
- Stax API Widget update, 487
- Stax application canonical MVC elements, 430
- Stax Concept Creator Widget, 494–495
- Stax End User Details screen for Dojo Console Widget, 262
- Stax End User Details screen for user input form, 257
- Stax operational baseline, 424
- Stax RIAPage, 425
- StaxConceptCreatorvA.kon loaded in Widget Engine, 485
- style sheet applied to DOM, 4
- tag cloud, 392–393, 447
- tagging infrastructure architecture, 395
- Tags and Photos relationship, 402
- tags table entries, 435
- traditional Web model, 122
- TurboGears Health History design, 183
- user created successfully in form validation, 284
- Venkman after startup, 153
- W3C's validation service, 155
- Web page with no applied styles, 4
- WEBrick server run for server-backed ideaStax Editor, 349
- Widget Converter Widget, 469–470
- Widget preferences dialog, 479

Figures (continued)

- XML document expressed as tree, 41
- XML document with page element, 40
- Yahoo and Google Traffic mashup combination, 111
- Yahoo's image search URL, 85
- YUI (Yahoo! User Interface) AutoCompleter, 202

FireBug debugger, features of, 154–155

FireBug in Layout Inspector mode, 154

FireBug plug-in for Firefox, using, 394

Firefox debugger, features of, 150–151

fixtures, role in Ruby on Rails test environment, 353

flags, defining in Dojo, 260, 526–529

Flickr Photo API

- accessing documentation for, 400
- extracting tags and photo identifiers from, 408–410
- setting up photo interface for, 411–415
- using in tagging infrastructure, 397

Flickr photo search result XML, 401–402

`flickrPhotos.py`, **411–412**

`float` **LSL data type, usage of, 503**

focus, maintaining (interview with 37Signals), 32

fold effect in `script.aculo.us`, argument for and description of, 370

“folksonomy,” 26, 391, 428. See also Web content for flow control keyword in LSL, description of, 509
`forceValidOption` **property of `ComboBox`**

Widget, default value and meaning of, 316

form before username specification, 263

form fields, adding validation to, 284–288

form submissions, 272

form usability. See client-side form usability; server-backed combo box; server-side form usability

form validation. See also client-side validation with Dojo; server-side form validation; validation

- adding submission of valid forms only, 268–272
- capturing user information for, 252
- client-side validation with JavaScript, 251
- server-side validation of, 250–251
- of user input form, 252–257

forms

- advantages to AJAX and JavaScript effects in, 297–298
- limitations to synchronous JavaScript in, 296–297
- `fps` **argument, using with `script.aculo.us` effects, 371**

frameworks for MVC
Ruby on Rails, 178–180

TurboGears framework in Python, 181–182

using with Java, 174–176

for Web applications, 176–178

frameworks in RIAs (interview with 37Signals), 32–33

`France.txt` **Hello file, content of, 133**

Fried, Jason and David Henemeier Hansson of 37Signals (interview), 31–34

from argument, using with `script.aculo.us` effects, 372

front controller execution in Java, 172

G

gardens. See “unwalled gardens;” “walled gardens”

generate script, using with Rails, 350

geocode, getting for Yahoo and Google Traffic mashup, 97–101

`GeocodeProxy.html`, **98–99**

`geocoder.php`, **101**

geocoding example, 98

`Germany.txt` **Hello file, content of, 133**

GET tests, creating in `IdeaStax`, 240

`get_concept_html` **method, using with `Stax`, 461**

`get_concepts_by_tag` **method, using with `Stax`, 461–462**

`getAllResponseHeaders()` **method of `XMLHttpRequest` object, description of, 73**

`.getAttribute(string)` **method of DOM element, use of, 60**

`.getElementById(String)` **method, use of, 59**

`.getElementsByTagName(String)` **method, use of, 59**

ghosting `script.aculo.us` Draggable option, description of, 330

`globals.py` **for Flickr photo interface, 415**

`GMap2` **function, using in Yahoo and Google Traffic mashup, 106**

Google. See also Yahoo and Google Traffic mashup
financial success of, 29
invisibility of, 10–11

Google Desktop Gadgets, description of, 468

Google Map API, 101

code sample, 109–110

lack of geocoding capability in, 107

Google Web Toolkit (GWT)

automatic serialization in, 214–215

benefits of, 207

developer tools in, 210–212

developing with, 208–210

drawbacks of, 208
 features of, 206–207
 look and feel of, 212–214
 versus RJS, 215–216

Google's Gmail interface, 372–373

GoogleYahooMashup.html, 112–115

greedy **Droppable** option, description of, 331

Grow **effect** in `script.aculo.us`, argument for and description of, 370

GWT (Google Web Toolkit). See also JavaScript

automatic serialization in, 214–215

benefits of, 207

developer tools in, 210–212

developing with, 208–210

drawbacks of, 208

features of, 206–207

look and feel of, 212–214

versus RJS, 215–216

GWT application running, 211

GWT hosted server and browser, 212

GWT packages, splitting up, 209

GWT project directory tree, 209

H

h1-h5 elements in XHTML, meaning of, 45

handle `script.aculo.us` **Draggable** option, description of, 330

handleCountryChange **method, defining for Hello World, 128**

handleFileDrop, using with **Stax API Widget, 487**

handleSubmit **function, defining for form validation, 270**

HEAD, **injecting SCRIPT tag into dynamically, 103–106**

head **element in XHTML, meaning of, 43**

HEAD section, placing scripting code in, 79

Health History design

in Java, 173–174

in Ruby on Rails MVC framework, 179–180

in TurboGears, 183

HealthDetailController, **relationship to MVC pattern, 167**

HealthDetailModel, **relationship to MVC pattern, 167**

HealthDetailView, **relationship to MVC pattern, 166–167**

HealthHistoryController, **relationship to MVC pattern, 167**

HealthHistoryModel, **relationship to MVC pattern, 167**

HealthHistoryView, **relationship to MVC pattern, 166–167**

height **property, use in CSS, 327**

“Hello, Avatar!,” displaying in Second Life, 502

Hello World example, 145

adding JavaScript event handler for, 128

creating asynchronous request for, 128

creating `helloFiles` directory for, 133–134

defining callback function for, 130–131

defining event handler function for, 128–130

defining JavaScript body for, 128

final version of, 145

processing server's response for, 131–132

Hello World RIA Widget. See also RIA Widgets

allowing customization with user preferences, 478–481

packing with **Widget Converter, 474–475**

replacing “Hello World!” message in, 475–477

setting up, 472–474

Hello World RIA Widget packed with Widget Converter, 475

Hello World Widget, 480

Hello World's web.xml, 138

helloCallback **implementation, 131**

helloFiles **directory, creating, 133–134**

HelloResponseServletv1.java, **139**

HelloResponseServletv2.java, **140–141**

helloResponsev1.php, **135–136**

helloResponsev2.php, **136–137**

HelloWorldv1.html, **126–127**

HelloWorldv3.html, **132–133**

HelloWorldv6.html, **144**

HelloWorldvA.kon, **472**

HelloWorldvB.kon **Widget, 481**

HelloWorld **page, 127**

helper class, defining for server-side form validation, 277–281. See also UserDataHelper

helper function, defining for client-side form usability, 306

helpers **subfolder in Rails, description of, 244, 445**

helpers/istaxHelper.rb **for tagging, 446**

Henemeier Hansson, David and Jason Fried of 37Signals (interview), 31–34

Hibernate framework, using with Java, 175

`hide` function in Prototype, description of, 368

hiding and showing elements, 368–369

Highlight effect in `script.aculo.us`, argument for and description of, 370

hover, considering for Droppable objects, 331

`hoverclass` Droppable option, description of, 331

HTML (hyperText markup language) versus XHTML, 47

HTML document without CSS file, 64

html element in XHTML, meaning of, 43

HTML elements, advisory about using in XHTML, 47

HTML tags, advisory about including in CSS files, 80

HTTP, exposing services over, 455

HTTP POST and GET tests, creating in ideaStax, 240

HTTP POST connection, using with Stax, 460

HTTP post, returning quote in response to, 475

HTTP requests

creating via `llHttpRequest` function in Web 2.0, 515–516

specifying for mashups, 74–75

`http_response` event in LSL, trigger associated with, 505

HTTP-based services, home growing, 459

ideaStax application, 222

creating tests for HTTP POST or GET in, 240

deconstructing Concepts, Cards, and Stacks in, 234–236

downloading, 230–231

implementing controllers in, 239

implementing views in, 237–239

levels of hierarchy in, 223

notification in, 374

partial render, 242

using Ruby JavaScript construct as partial renderer, 240–243

ideaStax Editor, 333. See also **client-only**

ideaStax Editor; drag and drop operations

components of, 333–335

design of, 333–335

implementation plan for, 336

including visual feedback in, 376

layout of, 356

with slides rearranged, 346

testing, 364

ideaStax hierarchy, 222

ideaStax interface, 223

IdeaStax Library

after loading photos, 342

before loading photos, 341

ideaStax Previewer modal window

amending `zoomInPreview` and `closePreviewMode` functions for, 385

completing, 386–387

creating, 378–379

creating semi-transparent “screen” for, 384

description of, 377

disposing, 383–384

disposing screen element in, 385

fetching, 381–382

layout calculations for, 380

making modal, 384–385

positioning, 379–381

sizing screen for, 384–385

transitioning in and out of, 383–384

zooming into, 383

if-else flow control keyword in LSL, description of, 509

IM (instant messaging) versus e-mail, 9

image concept, uploading to Stax API Widget, 488

`imageModel.py` for Flickr photo interface, 412–413

`img` element in XHTML, meaning of, 47

IMG tag, constructing for traffic report, 102–103

`index.rhtml`

for ideaStax, 237–238

for tagging, 439

initializePage function update, 345

inline JavaScript

attaching to Web pages, 55–56

execution of, 57

inner function, relocating in JSON, 91–92

input element in XHTML, meaning of, 47

instant messaging (IM) versus e-mail, 9

integer LSL data type, usage of, 503

integers, testing in Dojo, 526

`IntegerTextbox` Dojo validation Widget, function and test related to, 264

Internet Explorer, handling, 379–380

Internet resources

“The 20-Minute Wiki,” 29

Ajaxos desktop, 20

Apache, 71

- CherryPy framework, 181
 - “Creating a weblog in 15 minutes,” 29
 - Damn Small Linux, 21
 - displaying paragraphs with CSS in, 51
 - Dojo (AJAX edition), 258
 - Dojo JavaScript library, 188
 - FCKEditor, 21
 - FireBug plug-in for Firefox, 394
 - Flickr Photo API, 400
 - GWT (Google Web Toolkit) download, 208
 - Hibernate MVC framework, 175
 - ideaStax application download, 230
 - Jakarta Struts MVC frameworks, 176
 - JSF (JavaServer Faces) MVC frameworks, 175
 - JSMIn, 96
 - JSON (JavaScript Object Notation), 86
 - JSONparser, 91
 - Kid template language, 181
 - Lighttpd, 71
 - Markup Validation Service (W3C), 62
 - MashupFeed, 70
 - MochiKit JavaScript library, 181, 188
 - MochiKit JavaScript library download, 398
 - MySQL Connector JDBC library, 303
 - MySQLdb download, 396
 - ontology resource, 450
 - pattern catalog, 171
 - Prototype JavaScript library, 327
 - regular expressions, 279
 - Ruby on Rails MVC framework, 232, 348
 - Ruby on Rails MVC framework download, 431
 - script.aculo.us JavaScript library, 188
 - script.aculo.us JavaScript library, 327
 - Second Life environment download, 510
 - Shale MVC framework, 176
 - The Spring MVC framework, 175
 - SQLObject, 181
 - Stax download, 425, 428
 - tagging discussion, 423
 - tagging folksonomy, 428
 - Tapestry MVC framework, 176
 - Tomcat v5.5, 252
 - TurboGears, 181
 - validation service (W3C), 155
 - W3C’s Markup Validation Service, 62
 - Web server configuration to forward calls across domains, 108
 - widgEditor download, 428
 - Widget Converter Widget, downloading, 469
 - Widget gallery, 469
 - Yahoo! User Interface (YUI) JavaScript library, 188
 - Yahoo! Widget Engine, downloading, 469
 - Yahoo! Widget Engine Reference Guide, downloading, 469
 - Yahoo! Widgets Developer SDK, downloading, 469
 - interviews**
 - pertaining to 37Signals, 31–34
 - pertaining to Dojo Toolkit, 322–324
 - pertaining to Second Life environment, 519–522
 - `invalidClass` **Dojo validation Widget attribute, default value and meaning of, 265**
 - `invalidColor` **Dojo validation Widget attribute, default value and meaning of, 265**
 - `invalidMessage` **Dojo validation Widget attribute, default value and meaning of, 265**
 - IP address convention, testing in Dojo, 529**
 - `IpAddressTextbox` **Dojo validation Widget, function and test related to, 264**
 - `is12HourTime` **function in Dojo, explanation of, 528**
 - `is24HourTime` **function in Dojo, explanation of, 528**
 - `isCurrency` **function in Dojo, explanation of, 527–528**
 - `isEmailAddress` **function in Dojo, explanation of, 530**
 - `isEmailAddressList` **function in Dojo, explanation of, 530**
 - `isInteger` **function in Dojo, explanation of, 526**
 - `isIpAddress` **function in Dojo, explanation of, 529**
 - “island of automation,” creating with tools, 12**
 - `isNumberFormat` **function in Dojo, explanation of, 527**
 - `isRange` **function in Dojo, explanation of, 527**
 - `IsRealNumber` **function in Dojo, explanation of, 526–527**
 - `istax/index.html` **for tagging, 446–447**
 - `isText` **function in Dojo, explanation of, 526**
 - `isUrl` **function in Dojo, explanation of, 529–530**
 - `isValidDate` **function in Dojo, explanation of, 528**
 - `isValidTime` **function in Dojo, explanation of, 528**
- J**
- Jakarta Struts MVC frameworks, using with Java, 176**
 - Japanese currency convention, verifying in Dojo, 532**

Java

- components for implementing MVC in, 184
- design of Health History in, 173–174
- evaluating, 20
- MVC frameworks for, 174–176
- versus Ruby on Rails design, 180
- using MVC in, 170–176
- using regular expressions in, 279

Java Servlets. See also Web applications

- compiling and running for server-side validation, 282–284
- programming for RIAs, 138–142
- using in server-side form validation, 276–277

JavaScript. See also GWT (Google Web Toolkit); RJS (Ruby JavaScript)

- accessing DOM tree from, 59–60
- for Angela's Ristorante Web site, 67
- attaching as separate file, 56
- attaching to Web pages, 55–56
- binding to user actions, 57–59
- creating slide object from, 338–339
- features of, 55
- inspecting and setting element contents in, 60–61
- limitations to synchronous JavaScript in forms, 296–297
- page lifecycle events in, 57–58
- performing client-side form validation with, 251
- user-driven events in, 58–59
- using with RIAPage requests, 80–81

JavaScript body, defining for Hello World example, 128

JavaScript console

- debugging with, 152
- error displayed in, 85

JavaScript effects, advantages in forms, 297–298

JavaScript event handler

- adding for Hello World example, 128
- inclusion in AJAX, 125

JavaScript libraries. See also Prototype JavaScript library; script.aculo.us JavaScript library

- capabilities of, 189–191
- Dojo library, 188, 191–194
- inclusion in Web application frameworks, 176
- interview with Alex Russell, 323
- logging features of, 157
- MochiKit library, 188, 195–198
- properties of, 189

- quick reference for, 204–205
- reasons for use of, 186–188
- script.aculo.us library, 188, 198–201
- Yahoo! User Interface (YUI), 188
- YUI (Yahoo! User Interface), 201–202

JavaScript Object Notation (JSON)

- constructs of, 87
- creating in `Tagger.cgi`, 417–418
- `eval` function and security in, 92
- referencing fields in, 87–88
- relocating inner function in, 91–92
- using as alternative to XML, 86
- using in Yahoo and Google Traffic mashup, 104
- versus XML, 96

JavaScript usability enhancements

- in Dojo JavaScript library, 191
- in JavaScript libraries, 190
- in MochiKit JavaScript library, 195
- in script.aculo.us JavaScript library, 198
- in YUI (Yahoo! UI) JavaScript library, 201

JavaServer Faces (JSF) MVC framework, using, 175

`jojo.dom` functions, 307

JSF (JavaServer Faces) MVC framework, using, 175

JSMIn Web address, 96

JSON (JavaScript Object Notation)

- constructs of, 87
- creating in `Tagger.cgi`, 417–418
- `eval` function and security in, 92
- referencing fields in, 87–88
- relocating inner function in, 91–92
- using as alternative to XML, 86
- using in Yahoo and Google Traffic mashup, 104
- versus XML, 96

JSON deconstruction, 88

JSON example, deconstructing, 88–91

JSON requests, specifying for mashups, 75

JSONparser Web address, 91

`json.php`, 95

`jQueryExample.html`, 89–91

`jump` flow control keyword in LSL, description of, 509

K

`key` LSL data type, usage of, 503

keys, using with APIs, 456

Kid template language, relationship to TurboGears, 181–182
 .kon file extension, explanation of, 471

L

label element in XHTML, meaning of, 47
 land_collision event in LSL, trigger associated with, 505

land_collision_end event in LSL, trigger associated with, 505

land_collision_start event in LSL, trigger associated with, 505

layout manager in GWT, using, 213–214

layout.css used in form validation, 255–257

layouts in Ruby on Rails

constructing for server-backed ideaStax Editor, 356–357

description of, 355

left property, use in CSS, 327

li element in XHTML, meaning of, 45

Library

adding toggle functionality to, 369

in ideaStax Editor, 333, 335

loading contents of, 341

Library controller, building for server-backed ideaStax Editor, 362–363

Library controller for server-backed ideaStax Editor, 362–363

library partial, constructing for server-backed ideaStax Editor, 357–358

Lighttpd Web address, 71

Linden Scripting Language (LSL). See also Second Life environment

built-in states for, 505–506

flow control keywords in, 509

representing collections in, 504

statements and flow control in, 508–510

using with Second Life, 502

LINK directive, specifying style sheets with externally, 79

link element in XHTML

meaning of, 48

using, 50

link_message event in LSL, trigger associated with, 505

list LSL data type, usage of, 503–504

listen event in LSL, trigger associated with, 505

listenOnKeypress Dojo validation Widget attribute, default value and meaning of, 265

Listings. See also code samples

acts_as_taggable.rb, 444

add_concept function for API, 462

add_image_concept function for API, 464

AddressLocale.java for form usability, 300–301

addSlideToTimeline function for client-only ideaStax Editor, 346

alert.js, 56

attribute coded as element, 38

card_controller for tagging, 440

_cardlist.rhtml, 441

card.rb for ideaStax, 234

cardship.rb for ideaStax, 235

Chat_Catcher LSL for Second Life, 511–512

closePreviewMode function for ideaStax

Previewer modal window, 385

concept_controller.rb in ideaStax, 239

concept.rb for ideaStax, 234

concept.rb for tagging, 434

containment diagram for Angela's Ristorante, 63

create_cardship.rb for ideaStax, 236

create_concepts.rb for Ruby on Rails, 233

create_taggings.rb, 433

create_tags.rb, 433

create.rhtml for ideaStax, 240

CSS applied to XHTML documents, 53

CSS class-based style, 52

CSS file added to XHTML document for Angela's Ristorante, 64–66

CSS instructions inside header of XHTML document, 50

CSS style based on element's ancestry, 53

CSS styling for ideaStax Previewer modal window, 379

dbHandler.py for Flickr photo interface, 413–414

DBLookup.py for Tag Cloud application, 418–420

default LSL script for Second Life, 502

developer key for API, 456

div element, 52

DOM element overridden with ID of titlebar, 52

draggable slides loaded into photo library, 340–341

DynamicScripting.html, 104–105

Listings (continued)

Listings (continued)

- Editor controller for server-backed ideaStax Editor, 361–362
- Editor view for server-backed ideaStax Editor, 359–360
- Flickr photo search result XML, 401–402
- flickrPhotos.py, 411–412
- GeocodeProxy.html, 98–99
- geocoder.php, 101
- globals.py for Flickr photo interface, 415
- Google Map API, 109–110
- GoogleYahooMashup.html, 112–115
- GWT module file, 209
- GWT page, 210
- Hello World's web.xml, 138
- HelloResponseServletv1.java, 139
- HelloResponseServletv2.java, 140–141
- helloResponsev1.php, 135–136
- helloResponsev2.php, 136–137
- HelloWorldv1.html, 127
- HelloWorldv3.html, 132–133
- HelloWorldv6.html, 144
- HelloWorldvA.kon, 472
- HelloWorldvB.kon Widget, 481
- helpers/istaxHelper.rb for tagging, 446
- ideaStax Editor layout, 356
- ideaStax Editor template, 334
- ideaStax Previewer modal window layout calculations, 380
- image and link element in XHTML, 47
- imageModel.py for Flickr photo interface, 412–413
- index.rhtml for ideaStax, 237–238
- index.rhtml for tagging, 439
- initializePage function update, 345
- inline JavaScript, 55
- istax/index.html for tagging, 446–447
- JavaScript attached as separate file, 56
- JavaScript for Angela's Ristorante Web site, 67
- JavaScript in HelloWorldv2.html, 130
- json.php, 95
- jQueryExample.html, 89–91
- layout.css used in form validation, 255–257
- Library controller for server-backed ideaStax Editor, 362–363
- library_list partial for server-backed ideaStax Editor, 358
- link to external CSS file, 50
- load_concept.rjs for ideaStax, 241
- load-db.bat for loading zip codes, 300
- log function, 159
- logging for Web page, 159–160
- logging region, 158
- LSL states and helper functions, 506–507
- onLoad and onUnload attributes, 58
- openPreviewMode function, 382
- page initialization for client-only ideaStax Editor, 343
- path_to_file function for API, 464
- photo fixture in Ruby on Rails, 353
- photo Model implementation for server-backed ideaStax Editor, 352
- photo partial for server-backed ideaStax Editor, 357
- photo schema definition for server-backed ideaStax Editor, 350
- photo tagging tables, 404–405
- PhotoGetter.cgi, 420–421
- PhotoGetter-produced Web page, 422
- Photos table schema, 404
- preview container for ideaStax Previewer modal window, 378
- preview partial for ideaStax Previewer modal window, 384–385
- quotation generator in Web 2.0 for Second Life, 515–516
- quotation server in Web 2.0, 517
- Rails database settings for server-backed ideaStax Editor, 349
- sanitize_filename function for API, 464
- save_file function for API, 463
- sendNotification function, 373
- serializable object in GWT, 215
- sizeToScreen function for ideaStax Previewer modal window, 381
- slide added to Viewer for client-only ideaStax Editor, 343
- slide Model implementation for server-backed ideaStax Editor, 352
- slide object created from JavaScript, 338–339
- slide schema definition for server-backed ideaStax Editor, 351
- slide structure for ideaStax Editor, 335
- stack.rb for ideaStax, 235
- Stax Rails layout default.rhtml, 230–231, 429
- StaxConceptCreatorvA.kon Widget, 483–484
- StaxConceptCreatorvB.kon Widget, 491–493

- style of element in XHTML overridden, 51
 - TableCreator.py for tags and photos, 406–407
 - Tagger.cgi, 418
 - Tagger.js for tag cloud, 415–416
 - tagging.html, 392–393
 - Tags table schema, 405
 - text organized in XHTML, 46
 - _thumbnail.rhtml for ideaStax, 240–241
 - _thumbnail.rhtml for tagging, 441
 - Timeline controller for server-backed ideaStax Editor, 363–364
 - timeline Model implementation for server-backed ideaStax Editor, 352
 - timeline_list partial for server-backed ideaStax Editor, 358–359
 - toggle functionality for Library contents, 369
 - user model object User.java for server-side form validation, 274–275
 - UserDataHelperv2.java update, 285–288
 - user-driven events in JavaScript, 58
 - UserInputv2.html (nonvalidating), 252–255
 - UserInputv5.html for client-side validation, 266–268
 - UserManagerServletv1.java for server-side form validation, 276–277
 - Web 2.0 quotation generator for Second Life, 515–516
 - web.xml file for server-side form validation, 277
 - XHTML document with inline JavaScript, 56
 - XHTML slide duplication, 342–343
 - XHTML tag type overridden by CSS, 41
 - XML describing person, 37
 - XML document in book publisher's database, 39–40
 - XML tags collapsed, 38
 - XML translated into XHTML, 44
 - XMLQueryExample.html, 76–80
 - xmlQuery.php, 95
 - XMLHttpRequest data returned as JSON, 86
 - XMLHttpRequest data returned as XML, 75
 - zip_codes.sql MySQL script, 299–300
 - ZipCodeAnalyzerv1.java, 301–302
 - ZipManagerServletv1.java, 304
 - ZipManagerServletv2.java, 312–313
 - zoomInPreview function for ideaStax Previewer modal window, 383
 - list.rhtml **partial, creating for server-backed ideaStax Editor, 357**
 - llHttpRequest **function in Web 2.0, 515–516**
 - load property, using with dojo.io.bind function, 290**
 - load_concept.rjs for ideaStax, 241**
 - load-db.bat for loading zip codes, 300**
 - log, adding message to, 159**
 - log **function, example of, 159**
 - logger, example of, 158–160**
 - logging**
 - alert(String) function used in, 158
 - in Dojo JavaScript library, 192
 - integrating into applications, 156–157
 - in JavaScript libraries, 190
 - in MochiKit JavaScript library, 195
 - in script.aculo.us JavaScript library, 198
 - in YUI (Yahoo! UI) JavaScript library, 201
 - logging div, adding to Web page dynamically, 160–161**
 - logging messages, 160**
 - lookupCities **method for combo box, 311–312**
 - code samples, 311–312
 - Lowercase **Dojo validation Widget attribute, default value and meaning of, 265**
 - LSL (Linden Scripting Language). See also Second Life environment**
 - built-in states for, 505–506
 - flow control keywords in, 509
 - representing collections in, 504
 - statements and flow control in, 508–510
 - using with Second Life, 502
 - LSL data types, using in Second Life environment, 503–504**
 - LSL multi-state script in action, 508**
 - LSL New Script template, 510**
 - LSL scripts, structure of, 509**
 - LSL states and helper functions, 506–507**
- ## M
- Magic Cap environment, features of, 498–500**
 - Magic Cap interface, 498–499**
 - Map API, using in Yahoo and Google Traffic mashup, 106–110**
 - markup languages**
 - CSS (cascading style sheets), 36, 48–54
 - JavaScript, 54–61
 - XHTML (eXtensible HyperText Markup Language), 36, 42–48
 - XML (eXtensible Markup Language), 35

Markup Validation Service, running XHTML code through, 62

MashupFeed Web address, 70

mashups. See also proxies; XML output; Yahoo and Google Traffic mashup combination

creation of, 69–70

definition of, 69

in JavaScript, 81

interview with 37Signals, 33–34

Yahoo! service example, 73–74

maxLength property of ComboBox Widget, default value and meaning of, 316

metaverses, examples of, 500–502

metered APIs, features of, 455–458

metering developer API access, 457

method property, using with dojo.io.bind function, 290

migrate task in Rake, running, 354

mimetype property, using with dojo.io.bind function, 290

missingClass Dojo validation Widget attribute, default value and meaning of, 265

missingMessage Dojo validation Widget attribute, default value and meaning of, 265

MochiKit inline logging pane, 196

MochiKit JavaScript library

advantages of, 197

capabilities of, 195–198

description of, 188–189

disadvantages of, 198

quick reference for, 204

relationship to TurboGears, 181

using in tagging infrastructure, 398–400

MochiKit logger, 157

modal preview, 386

modal window

amending zoomInPreview and close PreviewMode functions for, 385

completing, 386–387

creating, 378–379

creating semi-transparent “screen” for, 384

description of, 377

disposing, 383–384

disposing screen element in, 385

fetching, 381–382

layout calculations for, 380

making modal, 384–385

positioning, 379–381

sizing screen for, 384–385

transitioning in and out of, 383–384

zooming into, 383

mode property of ComboBox Widget, default value and meaning of, 316

Model 1 architecture in Java, description of, 170

Model 2 architecture in Java

controller in, 171

description of, 170

model in, 172–173

view in, 171–172

model in MVC for Model 2 Java architecture, 172–173

model object and command, comparing in Java and Ruby on Rails, 180

Model objects, implementing for server-backed ideaStax Editor, 351–353

model responsibilities in MVC pattern, 167

model validation framework, inclusion in Web application frameworks, 177

models subfolder in Rails, description of, 244, 445

Model-View-Controller (MVC) pattern

advantages of, 168, 170

as aggregate pattern, 171

disadvantages of, 169–170

features of, 165–167

origins of, 164–165

using in Java, 170–176

in Web applications, 167–168

monetary value

testing in Dojo, 527–528

verifying in Dojo, 531

money event in LSL, trigger associated with, 505

mouse handlers, enhancing for Stax API Widget, 488–489

Move effect in script.aculo.us, argument for and description of, 370

MoveBy effect in script.aculo.us, argument for and description of, 370

movement grid, considering for Draggable objects, 329

moving_end event in LSL, trigger associated with, 505

moving_start event in LSL, trigger associated with, 505

Mozilla DOM Inspector debugger, features of, 151–152. See also DOM (document object model)

Mozilla-based browsers, handling, 379–380**MVC (Model-View-Controller) pattern, 165**

- advantages of, 168, 170
- as aggregate pattern, 171
- disadvantages of, 169–170
- features of, 165–167
- origins of, 164–165
- using in Java, 170–176
- in Web applications, 167–168

MVC architecture in TurboGears, 182**MVC frameworks**

- Ruby on Rails, 178–180
- TurboGears framework in Python, 181–182
- using with Java, 174–176
- for Web applications, 176–178

MVC implementation quick reference, 184**MVC Model 1 architecture in Java, description of, 170****MVC Model 2 architecture in Java**

- controller in, 171
- description of, 170
- model in, 172–173
- view in, 171–172

MVC pattern

- high-level class diagram of, 166
- in Java's Model 2 architecture, 170
- variation of, 168

MySQL

- in tagging infrastructure, 430–431
- using, 231
- using in tagging infrastructure, 396

MySQL 4.0.26, downloading for use with server-side forms, 299**MySQL Connector JDBC library, downloading, 303****MySQLdb, using in tagging infrastructure, 396–397****N****namespace support**

- in Dojo JavaScript library, 191
- in JavaScript libraries, 189
- in MochiKit JavaScript library, 195
- in script.aculo.us JavaScript library, 198
- in YUI (Yahoo! UI) JavaScript library, 201

The Network Is the Computer pattern, 16–17**network-centric nature of RIAs, 21–23**

- `no_sensor` event in LSL, trigger associated with, 505

- `.nodeName` property of DOM element, use of, 60
- `.nodeValue` property of DOM element, use of, 60
- `not_at_rot_target` event in LSL, trigger associated with, 505
- `not_at_target` event in LSL, trigger associated with, 506
- notification in `ideaStax`, 374
- notifications example, 372–374
- number format, testing in Dojo, 527

O

- `object_rez` event in LSL, trigger associated with, 506
- objects, requesting by tags, 436
- Observer class, role in MVC pattern, 166
- `ol` element in XHTML, meaning of, 45
- `on_rez` event in LSL, trigger associated with, 506
- `onAbort` attribute in JavaScript, meaning of, 57
- `onBlur` user-driven event in JavaScript, meaning of, 58
- `onChange`
 - sortable list option, 332
 - user-driven event in JavaScript, 58
- `onClick` user-driven event in JavaScript, meaning of, 58
- `onDbClick` user-driven event in JavaScript, meaning of, 58
- `onDragDrop` event handler, using with `Stax API Widget`, 486
- `onDrop` callback, using with `Droppable` class, 375–376
- `onDrop` `Droppable` option, description of, 331
- `onError` attribute in JavaScript, meaning of, 57
- `onFocus` user-driven event in JavaScript, meaning of, 58
- `onHover` callback, using with `Droppable` class, 375–376
- `onHover` `Droppable` option, description of, 331
- `onLoad` action
 - modifying for Hello World RIA Widget, 479
 - for `Stax API Widget` drag and drop, 486–487
- `onLoad` attribute in JavaScript, meaning of, 57
- `onload` function
 - using in form validation, 270
 - using with `RIAPage` requests, 84
- only sortable list option, description of, 332

`onMouseDown` **handler, enhancing for Stax API Widget, 489**

`onMouseDown` **user-driven event in JavaScript, meaning of, 58**

`onMouseMove` **user-driven event in JavaScript, meaning of, 58**

`onMouseOut` **user-driven event in JavaScript, meaning of, 58**

`onMouseOver` **user-driven event in JavaScript, meaning of, 58**

`onMouseUp` **handler, enhancing for Stax API Widget, 488–489**

`onMouseUp` **user-driven event in JavaScript, meaning of, 58**

`onreadystatechange` **function, 91**

`Onreadystatechange` **property of XMLHttpRequest object, description of, 72**

ontology, relationship to semantic tagging, 450

`onUnload` **attribute in JavaScript, meaning of, 57**

`onUpdate` **sortable list option, description of, 332**

`Opacity` **effect in script.aculo.us, argument for and description of, 370**

Open Widget dialog box, 474

opening bracket (<), using in XML tags, 37

`open` (`method`, `URL`) **method of XMLHttpRequest object, description of, 73**

`open` (`method`, `URL`, `async`) **method of XMLHttpRequest object, description of, 73**

`open` (`method`, `URL`, `async`, `userName`) **method of XMLHttpRequest object, description of, 73**

`open` (`method`, `URL`, `async`, `userName`, `password`) **method of XMLHttpRequest object, description of, 73**

`openPreviewMode` (`id`) **function, creating, 381**

operating systems, relationship to applications, 12

options hash, using with Draggable objects, 329

OS killers, RIAs as, 19–20

`output=json`, **specifying as output format, 75**

`overlap`

- `Droppable` option, 331
- sortable list option, description of, 332

P

P **element in XHTML, meaning of, 45**

pages

- attaching CSS to, 50–51
- attaching JavaScript to, 55–56

- attaching JavaScript to as separate files, 56
- example of plastic Web pages, 4–6
- improving with style sheets, 6–7
- logging for, 159–160
- using divisions in, 7–8

paragraphs in Web sites, displaying with CSS, 51

`Parallel` **effect in script.aculo.us, argument for and description of, 370**

parameters, validating in server-side forms, 281

`.parentNode`

- method of DOM element, use of, 61
- property of DOM element, use of, 60

partials in Ruby on Rails

- constructing for server-backed ideaStax Editor, 357–359
- description of, 355

`path_to_file` **function for API, 464**

patterns, resource for, 171. See also counter-manifesto patterns; Model-View-Controller (MVC) pattern

phone number formats in U.S., verifying in Dojo, 531

photo fixture in Ruby on Rails, 353

photo interface, setting up for Flickr, 411–415

photo library, populating with draggable slides, 339–342

photo metadata, downloading for data mining, 400–401

photo Model implementation for server-backed ideaStax Editor, 352

Photo model, populating with sample data, 353

photo partial, constructing for server-backed ideaStax Editor, 357

`PhotoGetter.cgi`, **420–421**

PhotoGetter-produced Web page, 422

`photoLoader.py`, **408–410**

photos and tags, setting up data model for, 403–408

Photos table snapshot, 355

PHP code

- for managing proxy for `jQueryExample.html`, 95
- for managing proxy for `XMLQueryExample.html`, 95

Pictures database, populating, 408–415

plastic Web pages, example of, 4–6

plasticity

- examples of, 7
- significance of, 4

POST tests, creating in ideaStax, 240**preferences**

- adding to Stax API Widget, 490–494
- allowing customization of Widget with, 478–481

preview area of Stax, displaying Cards in, 425**Preview modal window**

- amending zoomInPreview and close
 - PreviewMode functions for, 385
- completing, 386–387
- creating, 378–379
- creating semi-transparent “screen” for, 384
- description of, 377
- disposing, 383–384
- disposing screen element in, 385
- fetching, 381–382
- layout calculations for, 380
- making modal, 384–385
- positioning, 379–381
- sizing screen for, 384–385
- transitioning in and out of, 383–384
- zooming into, 383

project directory structure, inclusion in Web application frameworks, 178**Prototype JavaScript library. See also JavaScript libraries**

- capabilities of, 326
- hiding and showing elements in, 368
- obtaining, 327

proxies. See also mashups

- use of, 108
- working with, 94–96
- working without, 96

Puff effect in script.aculo.us, argument for and description of, 370**Pulsate effect in script.aculo.us, argument for and description of, 370****Python**

- “Easter Egg,” 24
- philosophy of, 23
- TurboGears framework in, 181–183
- using in tagging infrastructure, 396

Q**query generator, creating in Web 2.0 for Second Life, 515–516****queue argument, using with script.aculo.us effects, 372****quick references**

- Dojo validation Widget attributes, 265
- for Dojo validation Widgets, 264
- for JavaScript libraries, 204–205
- for MVC implementation, 184

quotation server in Web 2.0, 517**quote**

- providing for Hello World RIA Widget, 475–477
- returning in response to HTTP post, 475
- updating on double-click for Hello World RIA Widget, 477

R**Rails MVC framework**

- application flow control model, 443
- components for implementing MVC in, 184
- controllers in, 360
- description of, 178
- design of Health History in, 179–180
- downloading, 348, 431
- downloading and setting up, 232–233
- fixtures in test environment, 353
- flow control model of, 243
- folder conventions, 445
- generate script in, 350
- Health History design, 180
- interpretation of URLs by, 360–361
- versus Java, 180
- migration requirements for, 351
- MVC architecture in, 179
- respond_to do loop in, 378
- Routes in, 360
- subfolders in, 244
- support for testing in, 353
- versus TurboGears, 181–183
- using in tagging infrastructure, 431–432
- views in, 355–360

rake command output, 354**Rake framework, using with server-backed ideaStax Editor, 353–355****range of string, testing in Dojo, 527**

- rangeClass Dojo validation Widget attribute, default value and meaning of, 265
- rangeMessage Dojo validation Widget attribute, default value and meaning of, 265
- readyState property of XMLHttpRequest object, description of, 72

real value number, testing in Dojo, 526–527

`RealNumberTextbox` **Dojo validation Widget, function and test related to, 264**

regular expressions

review of, 279

validating server-side forms with, 277–278

release behavior, considering for Draggable objects, 329

remote API, exposing, 15

`remote_data` **event in LSL, trigger associated with, 506**

`.removeAttribute(string)` **method of DOM element, use of, 61**

`.removeChild(element)` **method of DOM element, use of, 60**

render, partial render in ideaStax, 240–242

rendered form for tagging, 440

`respond_to` **do loop, using with ideaStax**

Previewer modal window, 378

`responseText` **property of XMLHttpRequest object**
description of, 72

using, 143

`responseXML` **property of XMLHttpRequest object**
using, 143

`responseXML` **property of XMLHttpRequest object, description of, 73**

restaurant example, comparing Web versus AJAX applications in, 118–120

`return` **flow control keyword in LSL, description of, 509**

`revert` **script.aculo.us Draggable option, description of, 330**

`revertEffect` **property, using with Draggable class, 375**

`revertEffect` **script.aculo.us Draggable option, description of, 330**

RIA development environments (interview with 37Signals), 33

RIA development styles versus desktop applications, 18

RIA objects, 347

RIA solutions, 19

RIA Widgets. See also Hello World RIA Widget; Stax Concept Creator Widget; Widgets

file structuring and packaging of, 470–471

unpacking Widgets with Widget Converter, 469–470

Widget specification file overview, 471

RIAPage requests

anonymous inner function used with, 82

`cleanupResponseString` function used with, 82

DOM tree used with, 82–83

extracting attribute information from, 83–84

JavaScript coding considerations, 80–81

page layout of, 84–85

`parseResult` function used with, 82

`readyState` function used with, 82

specifying, 76–80

styling issues related to, 80

RIAPages

description of, 71

dynamic generation of, 422

for Google Map API, 107

tag cloud example, 392–393

in Yahoo and Google Traffic mashup, 106

RIAs (Rich Internet Applications)

achieving large audiences for, 151

browser-centric nature of, 21

comparing to traditional Web, 7–8

creating “unwalled gardens” with, 13–17

development environment setup for, 126

as disruptive technology, 8–12

enhancing client application for, 142–145

evolution of, 26

focusing, 11–12

implementing with Draggable objects, 328–329

inherent collaborative nature of, 25–26

invisibility of, 9–12

as living labs, 27

making money from, 28–29

mind-altering impact of, 23–24

network-centric nature of, 21–23

as OS killers, 19–20

plasticity of, 4

programming Java Servlets for, 138–142

programming server-side PHP for, 135–138

running, 134

seductive nature of, 29–30

as software services, 24

tools for development and testing of, 71

up-to-date status of, 17–18

user-centered nature of, 25

XHTML in, 49

rich immersion environments. See also Second Life environment

Magic Cap, 498–500
Snow Crash, 500

Rich Internet Applications (RIAs)

browser-centric nature of, 21
comparing to traditional Web, 7–8
creating “unwalled gardens” with, 13–17
as disruptive technology, 8–12
evolution of, 26
focusing, 11–12
inherent collaborative nature of, 25–26
invisibility of, 9–12
as living labs, 27
making money from, 28–29
mind-altering impact of, 23–24
network-centric nature of, 21–23
as OS killers, 19–20
plasticity of, 4
seductive nature of, 29–30
as software services, 24
up-to-date status of, 17–18
user-centered nature of, 25

RJS (Ruby JavaScript). See also JavaScript

benefits of, 216
developing with, 217–218
drawbacks of, 216–217
features of, 215–216
versus GWT, 215–216
look and feel of, 218–220

root controller classes, using with TurboGears, 182

`rotation` LSL data type, usage of, 503

Routes, role in Ruby on Rails, 360**Ruby JavaScript construct, using as partial renderer, 240–243****Ruby on Rails MVC framework**

components for implementing MVC in, 184
controllers in, 360
description of, 178
design of Health History in, 179–180
downloading, 348, 431
downloading and setting up, 232–233
fixtures in test environment, 353
flow control model of, 243
`generate` script in, 350
Health History design, 180
interpretation of URLs by, 360–361
versus Java, 180

migration requirements for, 351
MVC architecture in, 179
`respond_to` do loop in, 378
Routes in, 360
subfolders in, 244
support for testing in, 353
versus TurboGears, 181–183
using in tagging infrastructure, 431–432
views in, 355–360

Ruby Web server, writing, 93

`run_time_permissions` event in LSL, trigger associated with, 506

Russell, Alex (Dojo Toolkit interview), 322–324**S**

`sanitize_filename` function for API, 464

`save_file` function for API, 463

Scale effect in `script.aculo.us`, argument for and description of, 371

schema, defining for server-backed ideaStax Editor, 350–351

screen, sizing for ideaStax Previewer modal window, 384–385

screen element, disposing in ideaStax Previewer modal window, 385

script element in XHTML, meaning of, 48

SCRIPT tag

injecting dynamically into HEAD, 103–106
using in Map API, 110

script.aculo.us JavaScript library. See also JavaScript libraries

advantages and disadvantages of, 200–201
autocompleter, 199
binding effects to, 374–377
capabilities of, 198–201, 326
combining effects in, 377
description of, 188–189
Draggable options for, 329–330
Droppable options for, 331
effects for flashing notifications, 373
effects in, 369–372
obtaining, 327
quick reference for, 204
using, 187
using `Builder` to create files with, 336–339
wiki for, 329

scripting code, placing in HEAD section, 79. See also dynamic scripting

`scroll` **sortable list option, description of, 332**

`ScrollTo` **effect in script.aculo.us, argument for and description of, 371**

`searchType` **property of ComboBox Widget, default value and meaning of, 317**

Second Life environment, 501. See also LSL (Linden Scripting Language); rich immersion environments

connecting to real life, 511–514

downloading, 510

economics of, 520

elements of, 510–511

features of, 500–503

immersive qualities of, 521

interview pertaining to, 519–522

LSL data types for, 503–504

social and community aspects of, 520

states in, 505–508

Web 2.0 demonstration of, 515–517

`select` **element, updating for client side of combo box, 318**

`self.up` and `self.down` **steps, using with Ruby on Rails, 351**

semantic tagging, 449–450

`send(content)` **method of XMLHttpRequest object, description of, 73**

`sendNotification` **function, 373**

`sensor` **event in LSL, trigger associated with, 506**

serialization, handling by GWT, 207, 214–215

server response, processing for Hello World, 131–132

server side, implementing for combo box example, 311–314

server-backed combo box. See also client-side form usability

adding event handler for client side, 318

defining controller for, 312–314

defining model for, 311–312

description of, 309–311

implementing client side for, 314–321

implementing server side for, 311–314

specifying callback function for client side, 319–320

updating `onload` function for client side, 321

updating `select` element for client side, 318

using Dojo combo boxes with, 314–317

server-backed ideaStax Editor. See also client-only ideaStax Editor

building Editor controller for, 361–362

building Library controller for, 362–363

building Timeline controller for, 363–364

constructing Editor view for, 359–360

constructing library partial for, 357–358

constructing photo partial for, 357

constructing timeline partial for, 358–359

creating controller for, 360–364

creating Model for, 349–355

creating view for, 355–360

defining database schema for, 350–351

description of, 347–348

implementing Model objects for, 351–353

using Rake framework with, 353–355

using Ruby on Rails framework with, 348–349

server-side CGI, creating for tagged photo, 420–421

server-side form usability. See also client-side form usability

defining controller for, 303–305

defining model for, 300–303

implementing, 299

setting up database for, 299–300

server-side form validation. See also client-side validation with Dojo; form validation

asynchronous type of, 251

defining attributes for, 279–280

defining controller for, 276–277

defining helper class for, 277–281

defining method to validate single field for, 280–281

defining model for, 273–275

implementing, 272–273

implementing UserDataHelper for, 278–281

synchronous type of, 250

server-side PHP, programming for RIA, 135–138

server-side quote of day query handler, creating in Web 2.0, 517

server-side username, validating, 288–292

services, exposing over HTTP, 455

Servlets

compiling and running for server-side validation, 282–284

programming for RIAs, 138–142

using in server-side form validation, 276–277

- `.setAttribute(string1, string2)` method of DOM element, use of, 61
- `setRequestHeader(label, value)` method of XMLHttpRequest object, description of, 73
- Shake effect in `script.aculo.us`, argument for and description of, 371
- Shale Framework, using with Java, 176
- `show` function in Prototype, description of, 368
- showing and hiding elements, 368–369
- Shrink effect in `script.aculo.us`, argument for and description of, 371
- slide Model implementation for server-backed ideaStax Editor, 352
- SlideDown effect in `script.aculo.us`, argument for and description of, 371
- slides, creating dynamically with `script.aculo.us` Builder, 336–339
- SlideUp effect in `script.aculo.us`, argument for and description of, 371
- Smalltalk, relationship to MVC pattern, 164
- `snap` `script.aculo.us` Draggable option, description of, 330
- Snow Crash environment, features of, 500
- SOAP (Simple Object Access Protocol), choosing as API solution, 454
- social security number formats in U.S., verifying in Dojo, 531
- software subscription model (interview with 37Signals), 31–32
- `sort` Action, using with Timeline controller, 364
- Sortable behavior, implementing for client-only ideaStax Editor, 345–346
- sortable lists, defining, 332
- Sortable objects, adding visual effects with, 377
- sortable Timeline, implementing for client-only ideaStax Editor, 344–346
- `Spain.txt` Hello file, content of, 133
- `span` element in XHTML
 - explanation of, 43
 - use of, 44
- The Spring MVC framework, using with Java, 175
- SQLObjects, relationship to TurboGears, 181–182
- square brackets ([]), using in Dojo, 260, 527
- Squish effect in `script.aculo.us`, argument for and description of, 371
- Stack, Card, and Concept in ideaStax, 229
- Stack model, explanation of, 235
- Stack panel, contents of, 424
- `stack.rb` for ideaStax, 235
- Stacks in ideaStax, 229
 - description of, 222–223
 - verifying, 235–236
- standards compliance, using Firefox for, 150
- `starteffect` property, using with Draggable class, 375
- `starteffect` `script.aculo.us` Draggable option, description of, 330
- state abbreviation, verifying in Dojo, 531
- `state` flow control keyword in LSL, description of, 509
- `state_entry` event in LSL, trigger associated with, 506
- `state_exit` event in LSL, trigger associated with, 506
- stateless Web API, explanation of, 456
- states, using in Second Life environment, 505–508
- Status property of XMLHttpRequest object, description of, 73
- `statusText` property of XMLHttpRequest object, description of, 73
- Stax
 - containment model for, 424
 - downloading, 425
- Stax API
 - documentation sample for, 459–462
 - writing documentation for, 459
- Stax API Widget update, 487
- Stax application canonical MVC elements, 430
- Stax code, downloading for tagging, 428–430
- Stax Concept Creator Widget, 494–495. *See also* RIA Widgets
 - adding file drag and drop to, 486–487
 - adding user preferences to, 490–494
 - enhancing `onMouseDown` handler for, 489
 - enhancing `onMouseUp` handler for, 488–489
 - incorporating server response for, 489–490
 - setting up, 483–485
 - uploading image concept to, 488
- Stax Design Center, 424–426
- Stax End User Details screen for Dojo Console Widget, 262
- Stax End User Details screen for user input form, 257
- `stax` folder, contents of, 348
- Stax method call, making, 460

Stax methods

- `add_image_concept`, 460–461
- `get_concept_html`, 461
- `get_concepts_by_tag`, 461–462

Stax operational baseline, 424

Stax Rails layout default.rhtml, 429

Stax RIAPage, 425

Stax server, uploading concept for Stax API Widget to, 488–490

`StaxConceptCreatorVA.kon` **loaded in Widget Engine**, 485

`StaxConceptCreatorVA.kon` **Widget**, 483–484

`StaxConceptCreatorVB.kon` **Widget**, 491–493

`string` **LSL data type, usage of**, 503

string range, testing in Dojo, 527

strong element in XHTML, meaning of, 45

style added to Web page, 6

style sheet applied to DOM, 4

style sheets

- applying to DOM, 5
- creating better Web pages with, 6–7
- specifying externally, 79

styles

- implementing tag cloud with, 415–417
- specifying with CSS, 51

`submitCallback` **function**, 283, 283–284

Swing- versus XHTML-based development, 213

`SwitchOff` **effect in script.aculo.us, argument for and description of**, 371

`sync` **argument, using with script.aculo.us effects**, 372

synchronous form validation, explanation of, 250

synchronous JavaScript, limitations in forms, 296–297

T

`table` **element in XHTML, meaning of**, 45

`TableCreator.py` **for tags and photos**, 406–407

tag argument, using with Stax API Widget, 489–490

Tag Cloud application

- database lookups in, 418–420
- size of tag fonts in, 418

tag clouds

- adding, 443–448
- explanation of, 391–393
- implementing with styles, 415–417

tag editing area, location of, 438

tag pairs, exceptions to, 38. *See also* XML tags

`tag` **sortable list option, description of**, 332

tagged photo, server-side CGI for, 420–421

`Tagger.cgi`, **JSON created in**, 417–418

tagging

- discussion of, 423
- problem associated with, 390–391
- validating via console command, 434–437

tagging application, 391–395

`Tagging` **class, using**, 437

tagging folksonomy, 428

tagging infrastructure

- description of, 395–396
- downloading Stax code for, 428–430
- implementing cardlist renderer for, 441–443
- implementing controllers and views in, 437–441
- installing `acts_as_taggable` plugin for, 432–434
- semantic tagging, 449–450
- using Apache Web server in, 397–398
- using Flickr Photo API in, 397
- using MochiKit in, 398–400
- using MySQL in, 396, 430–431
- using MySQLdb in, 396–397
- using Python in, 396
- using Ruby on Rails framework with, 431–432

tagging infrastructure architecture, 395

`tagging.html`, 392–393

`.tagName` **property of DOM element, use of**, 60

tags

- extracting from Flickr, 408–409
- requesting objects by, 436

tags and photos

- setting up data model for, 403–408
- relationship between, 402

tags table entries, 435

Tapestry MVC framework, using with Java, 176

`td` **element in XHTML, meaning of**, 45

technology development, time involved in, 8

test framework, inclusion in Web application frameworks, 177

test-driven development, considering, 27

testing, support in Ruby on Rails MVC framework, 353

text string length, testing in Dojo, 526

`textarea` **input box, transferring text from**, 61

- `_thumbnail.rhtml`
 - for `ideaStax`, 240–241
 - for tagging, 441
- time formats, testing in Dojo, 528**
- Timeline controller, building for server-backed `ideaStax` Editor, 363–364**
- Timeline in `ideaStax` Editor, description of, 333, 335**
- timeline Model implementation for server-backed `ideaStax` Editor, 352**
- `timeline` **partial, constructing for server-backed `ideaStax` Editor, 358–359**
- `timeline_list` **partial for server-backed `ideaStax` Editor, 358–359**
- Timelines, creating with `ideaStax` Editor, 364**
- `timer` **event in LSL, trigger associated with, 506**
- `TimeTextbox` **Dojo validation Widget, function and test related to, 264**
- `title` **element in XHTML, meaning of, 43**
- `to` **argument, using with `script.aculo.us` effects, 372**
- `Toggle` **effect in `script.aculo.us`, argument for and description of, 371**
- `toggle` **function in Prototype**
 - description of, 368
 - using, 369
- Tomcat v5.5, downloading, 252**
- `Toolbox` **utility, using with TurboGears, 182**
- `top` **property, use in CSS, 327**
- `touch` **event in LSL, trigger associated with, 506**
- `touch_end` **event in LSL, trigger associated with, 506**
- `touch_start` **event in LSL, trigger associated with, 506**
- `tr` **element in XHTML, meaning of, 45**
- traditional Web model, 122. See also Web applications**
- `transition` **argument, using with `script.aculo.us` effects, 371**
- Trim Dojo validation Widget attribute, default value and meaning of, 265**
- TurboGears framework in Python**
 - components for implementing MVC in, 184
 - design of Health History in, 183
 - features of, 181–182
 - MVC architecture in, 182
 - versus Ruby on Rails, 181–183
- TurboGears Health History design, 183**
- U**
 - `ucFirst` **Dojo validation Widget attribute, default value and meaning of, 265**
 - `ul` **element in XHTML, meaning of, 45**
 - unit test framework**
 - in Dojo JavaScript library, 193
 - in JavaScript libraries, 191
 - in MochiKit JavaScript library, 197
 - in `script.aculo.us` JavaScript library, 200
 - in YUI (Yahoo! UI) JavaScript library, 202
 - “unwalled gardens.” See also “walled gardens”**
 - constraints in, 14–15
 - creating, 13–17
 - relationship to counter-manifesto, 15–17
 - `upload` **command, using with Stax API Widget, 489**
 - `Uppercase` **Dojo validation Widget attribute, default value and meaning of, 265**
 - URL formatting convention, testing in Dojo, 529–530**
 - `url` **property, using with `dojo.io.bind` function, 290**
 - URL routing and controller-view association, inclusion in Web application frameworks, 177**
 - URLs, interpretation by Ruby on Rails, 360–361**
 - `UrlTextbox` **Dojo validation Widget, function and test related to, 264**
 - U.S. phone number formats, verifying in Dojo, 531**
 - U.S. social security number formats, verifying in Dojo, 531**
 - U.S. zip code number formats, verifying in Dojo, 532**
 - `USA.txt` **Hello file, content of, 133**
 - user access to APIs, controlling, 455–456**
 - user actions, binding JavaScript to, 57–59**
 - user agnosticism, enabling, 16**
 - user created successfully in form validation, 284**
 - user experience as functional specification (interview with 37Signals), 33–34**
 - user information, capturing for form validation, 252**
 - user input form, validating, 252–257**
 - user interface components**
 - in Dojo JavaScript library, 192
 - in JavaScript libraries, 190
 - in MochiKit JavaScript library, 195
 - in `script.aculo.us` JavaScript library, 198
 - in YUI (Yahoo! UI) JavaScript library, 201
 - user model object User.java for server-side form validation, 274–275**

user objects in server-side forms, defining method for, 281

user preferences

adding to Stax API Widget, 490–494

allowing customization of Widget with, 478–481

`UserDataHelper`, **implementing for server-side**

form validation, 278–281. See also helper class

`UserDataHelperV2.java` **update, 285–288**

user-driven events, relationship to JavaScript, 58–59

`userformhelperv1.js`, **305–306**

`userform-helperv1.js` **file, creating, 268**

`UserFormv7.html`, **calling client-side Servlet from, 282–283**

`UserInputv2.html` **(nonvalidating), 252–255**

username, server-side validation of, 288–292

Username Field Validation Widget, adding for form validation, 262–263

`us.isCurrency` **function in Dojo, explanation of, 531**

`us.isGermanCurrency` **function in Dojo, verifying in Dojo, 532**

`us.isJapaneseCurrency` **function in Dojo, verifying in Dojo, 532**

`us.isPhoneNumber` **function in Dojo, explanation of, 531**

`us.isSocialSecurityNumber` **function in Dojo, explanation of, 531**

`us.isState` **function in Dojo, explanation of, 531**

`us.isZipCode` **function in Dojo, verifying in Dojo, 532**

`usPhoneNumberTextbox` **Dojo validation Widget, function and test related to, 264**

`UsSocialSecurityNumber` **Dojo validation Widget, function and test related to, 264**

`UsStateTextbox` **Dojo validation Widget, function and test related to, 264**

`UsZipTextbox` **Dojo validation Widget, function and test related to, 264**

V

`VaildationTextbox` **Dojo validation Widget, function and test related to, 264**

validateField function

arguments of, 280

calling, 270

creating for form, 269

`validateUsername` **function, adding, 290**

validation. See also form validation

adding to form fields, 284–288

functions in Dojo JavaScript library, 526–529

requirements for Cards in ideaStax, 235

of tagging via console command, 434–437

validation request, defining function for checking of, 283

validation service

features of, 155–156

running XHTML code through, 62

`validColor` **Dojo validation Widget attribute, default value and meaning of, 265**

`vector` **LSL data type, usage of, 503**

Venkman debugger, features of, 152–153

versions of applications, considering, 27

view composition inheritance, inclusion in Web application frameworks, 177

view responsibilities in MVC pattern, 166–167

view specification language, inclusion in Web application frameworks, 177

Viewer, adding slide to, 343

`view.rhtml` **file, creating for server-backed ideaStax Editor, 359**

views

constructing Editor view for server-backed ideaStax Editor, 359–360

creating for server-backed ideaStax Editor, 355–360

enhancing for client-side form validation, 282–284

enhancing for server-side username validation, 289–291

implementing in ideaStax, 237–239

implementing in tagging infrastructure, 437–441

in MVC for Model 2 Java architecture, 171–172

in Ruby on Rails MVC framework, 355–360

views subfolder in Rails, description of, 244, 445

visual effects

adding with `Sortable` objects, 377

binding to `script.aculo.us` actions, 374–377

in Dojo JavaScript library, 192

in JavaScript libraries, 190

in MochiKit JavaScript library, 196

in `script.aculo.us` JavaScript library, 199, 369–372

in YUI (Yahoo! UI) JavaScript library, 201

W

W3C's validation service, features of, 155–156

“walled gardens,” breaking down, 12–13. *See also*
“unwalled gardens”

Web, collaboration on, 26

Web 2.0

client-side query generator in, 515–516

demo of Second Life in, 515–517

server-side quote of day query handler in, 517

Web 2.0 design, benefit of, 30

Web 2.0 innovation (interview with 37Signals), 31

Web 2.0 Is a Cultural Phenomenon pattern, 17

Web 2.0 layering, 19

Web APIs, statelessness of, 456

**Web application frameworks, capabilities of,
176–178**

**Web applications. *See also* AJAX (Asynchronous
JavaScript and XML); Java Servlets**

compiling and running for form validation, 284

including Dojo JavaScript library in, 259

integrating logging into, 156–157

limitations on traditional approach, 122–123

MVC pattern in, 167–168

traditional approach toward, 121

using AJAX with, 123–125

**Web content, adding folksonomy to, 391. *See also*
content; “folksonomy”**

Web design, changes in, 8

**Web page with DOM elements, tag pairs, and JPEG
image, 5**

Web page with no applied styles, 4

Web pages

attaching CSS to, 50–51

attaching JavaScript to, 55–56

attaching JavaScript to as separate files, 56

example of plastic Web pages, 4–6

improving with style sheets, 6–7

logging for, 159–160

using divisions in, 7–8

Web programming, document focused nature of, 35

**Web server, inclusion in Web application
frameworks, 178**

**Web Services. choosing as API solutions, 454–455,
458–459**

**Web Services Description Language (WSDL),
explanation of, 458–459**

**Web sites. *See also* Angela's Ristorante wait-staff
portal**

“The 20-Minute Wiki,” 29

Ajaxos desktop, 20

Apache, 71

CherryPy framework, 181

“Creating a weblog in 15 minutes,” 29

Damn Small Linux, 21

displaying paragraphs with CSS in, 51

Dojo (AJAX edition), 258

Dojo JavaScript library, 188

FCKEditor, 21

FireBug plug-in for Firefox, 394

Flickr Photo API, 400

GWT (Google Web Toolkit) download, 208

Hibernate MVC framework, 175

ideaStax application download, 230

Jakarta Struts MVC frameworks, 176

JSF (JavaServer Faces) MVC frameworks, 175

JSMIn, 96

JSON (JavaScript Object Notation), 86

JSONparser, 91

Kid template language, 181

Lighttpd, 71

Markup Validation Service (W3C), 62

MashupFeed, 70

MochiKit JavaScript library, 181, 188

MochiKit JavaScript library download, 398

MySQL Connector JDBC library, 303

MySQLdb download, 396

ontology resource, 450

pattern catalog, 171

Prototype JavaScript library, 327

regular expressions, 279

Ruby on Rails MVC framework, 232, 348

Ruby on Rails MVC framework download, 431

script.aculo.us JavaScript library, 188

script.aculo.us JavaScript library, 327

Second Life environment download, 510

Shale MVC framework, 176

The Spring MVC framework, 175

SQLObject, 181

Stax download, 425, 428

tagging discussion, 423

tagging folksonomy, 428

Tapestry MVC framework, 176

Tomcat v5.5, 252

Web sites (continued)

TurboGears, 181
validation service (W3C), 155
W3C's Markup Validation Service, 62
Web server configuration to forward calls across domains, 108
widgEditor download, 428
Widget Converter Widget, downloading, 469
Widget gallery, 469
Yahoo! User Interface (YUI) JavaScript library, 188
Yahoo! Widget Engine, downloading, 469
Yahoo! Widget Engine Reference Guide, downloading, 469
Yahoo! Widgets Developer SDK, downloading, 469

weblog resource, 29

WEBrick server, running for server-backed ideaStax Editor, 349

web.xml file
for form usability, 305
for Hello World, 138
for server-side form validation, 277

while flow control keyword in LSL, description of, 509

widgEditor, downloading for ideaStax, 230, 428

Widget applications, examples of, 468

Widget Converter Widget, 469–470
downloading, 469
packing Hello World RIA Widget with, 474–475
unpacking Widgets with, 469–470

Widget Engine, visualizing Widgets in, 482

Widget gallery Web address, 469

Widget preferences dialog, 479

Widget specification, 471
adding `action` element to end of, 475–477
changes in format for, 473

Widgets. See also RIA Widgets
client interactions with, 482
relationship to GWT, 207
shift from elements to attributes in, 473
testing, 474
visualizing, 482

width property, use in CSS, 327

wiki for script.aculo.us JavaScript library, 329

window element, role in Hello World RIA Widget, 473

WSDL (Web Services Description Language), explanation of, 458–459

X

XHTML (eXtensible HyperText Markup Language)

avoiding HTML elements in, 47
describing text type in, 45–46
description of, 36
document setup in, 47–48
document structure of, 43–44
features of, 42
versus HTML, 47–48
in-text objects in, 46–47
order of elements in, 44
origins of, 43
in RIAs, 49
text structure of, 44–45

XHTML code, running through W3C Markup Validation Service, 62

XHTML documents

with inline JavaScript, 55–56

XHTML documents, placing CSS instructions in header of, 50

XHTML elements, applying CSS to, 53

XHTML slide duplication, 342–343

XHTML tag types, overriding with CSS, 51

XHTML- versus Swing-based development, 213

XML (eXtensible Markup Language)

attributes in, 38–39
elements and attributes in, 37–38
description of, 35
versus JSON, 96
purpose of, 36
relationship to DOM tree, 39–42
using JSON as alternative to, 86

XML document with page element, 40

XML documents

components of, 37
expressing as trees, 41
order of tags in, 37

XML output. See also mashups

specifying HTTP requests, 74–75
specifying JSON requests for, 75

XML tags, example of, 37. See also tag pairs

XMLHttpRequest

creating for Hello World, 128
inclusion in AJAX, 125
methods of, 73
properties of, 72–73

response properties of, 143

returning as XML, 75

`XMLQueryExample.html`, managing proxy for, 95

`xmlQuery.php`, 95

XMLHttpRequest data returned as JSON, 86

XY drop criteria, considering for `Droppable` objects, 330

Y

Yahoo and Google Traffic mashup. See also Google

combining, 110–116

getting geocode for, 97–101

getting traffic report from, 97–101

`SCRIPT` tag injected dynamically into `HEAD`,
103–106

using Map API in, 106–110

Yahoo and Google Traffic mashup combination, 111

Yahoo! service mashup, 73–74

Yahoo! User Interface (YUI) JavaScript library

advantages and disadvantages of, 203

capabilities of, 201–202

description of, 188–189

quick reference for, 204

Yahoo! Widget Engine, downloading, 469

**Yahoo! Widget Engine Reference Guide,
downloading, 469**

Yahoo Widgets, description of, 468

Yahoo! Widgets Developer SDK, downloading, 469

Yahoo's image search URL, 85

"yak shaving," 9–10, 17–18

**YAML (YAML Ain't Markup Language), defining Ruby
fixtures with, 353**

YUI (Yahoo! User Interface) JavaScript library

advantages and disadvantages of, 203

capabilities of, 201–202

description of, 188–189

quick reference for, 204

YUI AutoCompleter, 202

Z

**Z-Index drop criteria, considering for `Droppable`
objects, 330**

z-index property, use in CSS, 327

**zindex script.aculo.us `Draggable` option,
description of, 330**

**zip code number formats in U.S., verifying in Dojo,
532**

zip_codes.sql MySQL script, 299–300

ZipCodeAnalyzerv1.java, 301–302

ZipManagerServletv1.java, 304

ZipManagerServletv2.java, 312–313

**zoomInPreview function, amending for ideaStax
Previewer modal window, 385**

