

/THEORY/IN/PRACTICE

Ajax Design Patterns

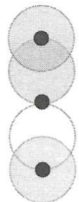
Creating Web 2.0 Sites with Programming and Usability Patterns

Michael Mahemoff

O'REILLY®

Petitioner Exhibit 1007
Page 1 of 15

Ajax Design Patterns



Michael Mahemoff

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

Ajax Design Patterns
by Michael Mahemoff

Copyright © 2006 Michael Mahemoff. All rights reserved. Printed in the United States of America.

Published by O'Reilly Media, Inc. 1005 Gravenstein Highway North, Sebastopol, CA 95472

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*safari.oreilly.com*). For more information, contact our corporate/institutional sales department: (800) 998-9938 or *corporate@oreilly.com*.

Editor: Simon St.Laurent

Cover Designer: Mike Kohnke

Production Editor: Mary Brady

Interior Designer: Marcia Friedman

Copyeditors: Mary Brady and Lydia Onofrei

Illustrators: Robert Romano and

Indexer: Julie Hawks

Jessamyn Read

Printing History:

June 2006: First Edition.

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. *Ajax Design Patterns* and related trade dress are trademarks of O'Reilly Media, Inc. Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

This work is licensed under the Creative Commons License Attribution 2.0. To view a copy of this License, visit <http://creativecommons.org/licenses/by/2.0/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California 94105-3013, USA.



This book uses RepKover™, a durable and flexible lay-flat binding.

ISBN: 0-596-10180-5
[M]

CONTENTS

	PREFACE	ix
Part One	INTRODUCTION	1
1	INTRODUCING AJAX	3
	<i>Ajax and the Usable Web</i>	3
	<i>The Rise of Ajax</i>	5
	<i>Ajaxifying the Web: The Story of Portals</i>	6
	<i>Webifying the Desktop: The Story of Office Applications</i>	9
	<i>Characteristics of Ajax Applications</i>	10
	<i>The Ajax Technologies</i>	14
	<i>Anatomy of a Server Call</i>	15
	<i>Ajax Trends</i>	17
	<i>Conclusions</i>	20
2	A PATTERN-LED TUTORIAL	21
	<i>Ajax Technologies in a Blink</i>	21
	<i>Ajaxifying a Web App: One Pattern at a Time</i>	27
	<i>Projects and Katas</i>	40
	<i>Conclusions</i>	41
3	AJAX DESIGN: PRINCIPLES AND PATTERNS	43
	<i>Desirable Attributes of Ajax Applications</i>	43
	<i>Designing for Ajax</i>	44
	<i>Ajax Patterns Overview</i>	47
	<i>Anatomy of a Pattern</i>	48
	<i>Ajax Patterns Demos</i>	54
	<i>Conclusions</i>	55
Part Two	FOUNDATIONAL TECHNOLOGY PATTERNS	57
4	AJAX APP	59
	<i>Ajax App</i>	59
5	DISPLAY MANIPULATION	69
	<i>Display Morphing</i>	69
	<i>Page Rearrangement</i>	78

6	WEB REMOTING	85
	<i>Web Service</i>	86
	<i>XMLHttpRequest Call</i>	91
	<i>IFrame Call</i>	108
	<i>HTTP Streaming</i>	113
	<i>On-Demand JavaScript</i>	122
7	DYNAMIC BEHAVIOR	133
	<i>User Action</i>	133
	<i>Scheduling</i>	143
8	EXTENDED TECHNOLOGIES	149
	<i>Richer Plugin</i>	149
Part Three	PROGRAMMING PATTERNS	159
9	WEB SERVICES	161
	<i>RESTful Service</i>	162
	<i>RPC Service</i>	177
	<i>Ajax Stub</i>	182
	<i>HTML Message</i>	186
	<i>Plain-Text Message</i>	191
	<i>XML Message</i>	195
	<i>JSON Message</i>	201
10	BROWSER-SERVER DIALOGUE	209
	<i>Call Tracking</i>	210
	<i>Periodic Refresh</i>	215
	<i>Submission Throttling</i>	221
	<i>Explicit Submission</i>	227
	<i>Distributed Events</i>	232
	<i>Cross-Domain Proxy</i>	243
11	DOM POPULATION	253
	<i>XML Data Island</i>	254
	<i>Browser-Side XSLT</i>	259
	<i>Browser-Side Templating</i>	265
12	CODE GENERATION AND REUSE	275
	<i>Server-Side Code Generation</i>	275
	<i>Cross-Browser Component</i>	283

13	PERFORMANCE OPTIMIZATION	289
	<i>Browser-Side Cache</i>	290
	<i>Predictive Fetch</i>	297
	<i>Guesstimate</i>	303
	<i>Multi-Stage Download</i>	310
	<i>Fat Client</i>	317

Part Four	FUNCTIONALITY AND USABILITY PATTERNS	327
------------------	---	------------

14	WIDGETS	329
	<i>Slider</i>	330
	<i>Progress Indicator</i>	336
	<i>Drilldown</i>	343
	<i>Data Grid</i>	350
	<i>Rich Text Editor</i>	355
	<i>Suggestion</i>	361
	<i>Live Search</i>	368
	<i>Live Command-Line</i>	376
	<i>Live Form</i>	383

15	PAGE ARCHITECTURE	389
	<i>Drag-And-Drop</i>	390
	<i>Sprite</i>	394
	<i>Popup</i>	401
	<i>Malleable Content</i>	407
	<i>Microlink</i>	416
	<i>Portlet</i>	421
	<i>Status Area</i>	427
	<i>Update Control</i>	431
	<i>Virtual Workspace</i>	436

16	VISUAL EFFECTS	445
	<i>One-Second Spotlight</i>	446
	<i>One-Second Mutation</i>	455
	<i>One-Second Motion</i>	462
	<i>Highlight</i>	468

17	FUNCTIONALITY	473
	<i>Lazy Registration</i>	474
	<i>Direct Login</i>	487
	<i>Host-Proof Hosting</i>	493
	<i>Timeout</i>	499

	<i>Heartbeat</i>	509
	<i>Unique URLs</i>	515
Part Five	DEVELOPMENT PATTERNS	531
18	DIAGNOSIS	533
	<i>Logging</i>	534
	<i>Debugging</i>	537
	<i>DOM Inspection</i>	540
	<i>Traffic Sniffing</i>	544
19	TESTING	549
	<i>Simulation Service</i>	550
	<i>Browser-Side Test</i>	554
	<i>Service Test</i>	558
	<i>System Test</i>	560
Part Six	APPENDICES	565
A	AJAX FRAMEWORKS AND LIBRARIES	567
B	SETTING UP THE CODE EXAMPLES	601
C	PATTERNS AND PATTERN LANGUAGES	605
D	REFERENCES	607
	INDEX	609

Widgets

THE WIDGETS PROVIDED BY STANDARD HTML—TEXT INPUTS, RADIOBUTTONS, AND SO ON—HAVEN'T changed much for ten years and are getting a little tired. So many people have been building their own widgets on top of the standard HTML offerings, and a few patterns have emerged. It's unlikely you'll need to build these widgets yourself, because library implementations are available for almost all of them. Nevertheless, it's worth it to consider when to use them and how to configure them.

The chapter starts with smaller, isolated widgets and builds up to more complex components. *Slider* is "the widget HTML forgot"—a typical slider widget so useful it's surprising it hasn't yet become standard HTML. *Progress Indicator* is a simple message or animation shown while waiting for a web remoting call. *Drilldown* is a control for selecting an element within a hierarchy.

Traditional HTML widgets are turbo-charged in the next couple of patterns. *Data Grid* is a "table on steroids," with tools for querying and restructuring that will be familiar to users of spreadsheets and database report tools. And *Rich Text Editor* is a "textarea on steroids," with options such as font size and color similar to those of a modern word processor.

Suggestion is similar to the traditional combo-box, a mixture between browsing and searching that relies on web remoting to locate a set of options against a partial text input. *Live Search* works similarly but shows search results rather than helping to complete a text field. It also works with nontext controls.

Finally, *Live Command-Line* and *Live Form* are higher-level patterns that tie together various widgets and patterns. The former is largely speculative and explores techniques to help people learn and use command-line interfaces within the Web. The latter is a more proven Ajax Pattern—a form that keeps changing in response to user input.

Slider



Continuous, Lever, Multiple, Range, Slider

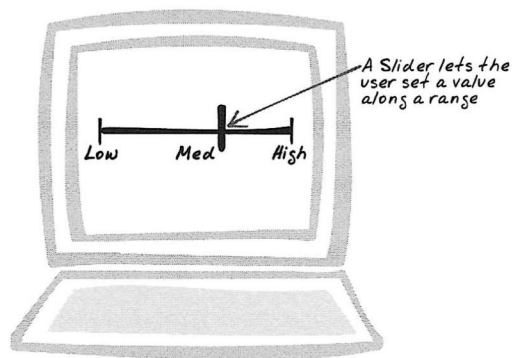


FIGURE 14-1. Slider

Goal Story

Stuart is rating his lecturer online. Each answer lies somewhere between “Strongly Disagree” and “Strongly Agree.” Unlike conventional surveys, the scale is continuous, because Stuart uses a Slider to set a precise value along the spectrum.

Problem

How can users specify a value within a range?

Forces

- Most data is constrained to reside within a range.
- Prevention is better than cure; the user interface should be designed such that the user is forced to choose a value within the range instead of the interface checking and providing an error message later on.

Solution

Provide a Slider to let the user choose a value within a range. The user drags a handle along one dimension to set the value. As a variant, two handles on the same Slider can allow the user to choose a range instead of a single value.

Standard HTML contains two related input controls: radiobuttons and selectors. Each of these lets you specify a value from a fixed list of choices. The biggest advantage of a Slider is that the data range can be continuous as well as discrete. Of course, “continuous” is an approximation, since you can only have as many choices as there are pixels in the Slider range. But with most Sliders, that means you have hundreds of unique values, which is continuous for most intents and purposes.

A Slider can also show discrete data by identifying several points along the range. When the user releases the handle, the Slider jumps to the nearest point. Why use a Slider when radio buttons and selectors already support discrete data? A Slider provides a better visual indication that the data resides in a spectrum. Also, it’s often faster because the user can click anywhere in the region as opposed to aiming precisely at a radiobutton or opening up a dialog box.

A further benefit of Sliders is their excellent support for comparing similar data. When several Sliders share the same range, they can be placed in parallel to show how the variables differ. For example, a product survey could ask questions such as “Were you happy with the price?” and “How easy was it to start using?” The answers lie on a different scale but ultimately map to the same range from “Unhappy” to “Happy.” Placing these horizontal rows in parallel helps the user stick to the same scale and compare each factor to the others.

Because Sliders aren’t standard controls, you’ll need to either use a library or roll your own. Typical control mechanisms include the following:

- Dragging the handle moves it directly.
- Clicking on either side of the handle moves it a little in that direction.
- While the Slider has keyboard focus, pressing left and right arrows move it in either direction. It’s sometimes useful to offer an accelerator key such as Shift, which, held down at the same time as the arrow, speeds up the handle movement. Keyboard controls are particularly important when the control is part of a form.
- Receiving notifications from external sources. The Slider is sometimes synchronized with another object, so if that object changes, the Slider must update too.

A typical implementation separates the scale from the handle. The scale consists of a line, possibly with some notches and labels. The handle is usually an image, with the `zIndex` property set to place it “in front of” the main Slider. There are several event handlers required to support all of the mechanisms above, and the Slider handle itself is manipulated using techniques discussed in *Drag-And-Drop* (Chapter 15). In addition, movements will often trigger other activity, such as an *XMLHttpRequest Call* or a change to another page element.

Decisions

How will you orient the Slider?

There are two options for orientation: horizontal or vertical. Following are a few considerations:

Data type

Sometimes the nature of the data dictates which option is more logical. For instance, Google Maps uses a vertical Slider for zoom, which corresponds to a feeling of flying toward and away from the map as you zoom in and out.

Layout

Aesthetic appearance and space conservation are important. Many forms will feature horizontal Sliders because they fit well underneath questions. Vertical Sliders would lead to a lot of whitespace.

Proximity

Where the Slider controls something else on the page, you'll probably want to place them near each other, which might dictate orientation.

Comparison

As mentioned earlier in the "Solution," Sliders work well when placed in parallel, which means a common orientation.

What scale will you use?

There are quite a few ways to present a variable, and the choice will depend on the nature of the data and what users are comfortable with. Examples include:

- Qualitative descriptions ("Low", "High"). This might seem suited only to discrete ranges, but the labels can also be used as markers within a continuous range.
- Absolute values.
- Percentages (typically ranging from 0 to 100).

How will you present the scale?

There are various strategies for presenting the scale:

- Provide no labels; rely on context and conventions. For instance, users—at least in western countries—usually assume Sliders increase in value to the right.
- Provide just a label at either end.
- Provide labels at several points along the range.

The labels are usually shown alongside the Slider, but to conserve space, you can sometimes show them inside it.

Real-World Examples

Yahoo! Mindset

A product of Yahoo Labs, Yahoo! Mindset (<http://mindset.research.yahoo.com/>) lets you tweak search results with an unusual type of Slider that ranges from “shopping” on one end to “researching” on the other (Figure 14-2). Pull the “Ajax” results towards “shopping” and you’ll see links to cleaning products and football. Pull it to the right and you’ll see some information about web design (and, it must be said, more football). Also of note: the Slider is “Live”—each time you change it, the results are automatically updated via an *XMLHttpRequest Call*.

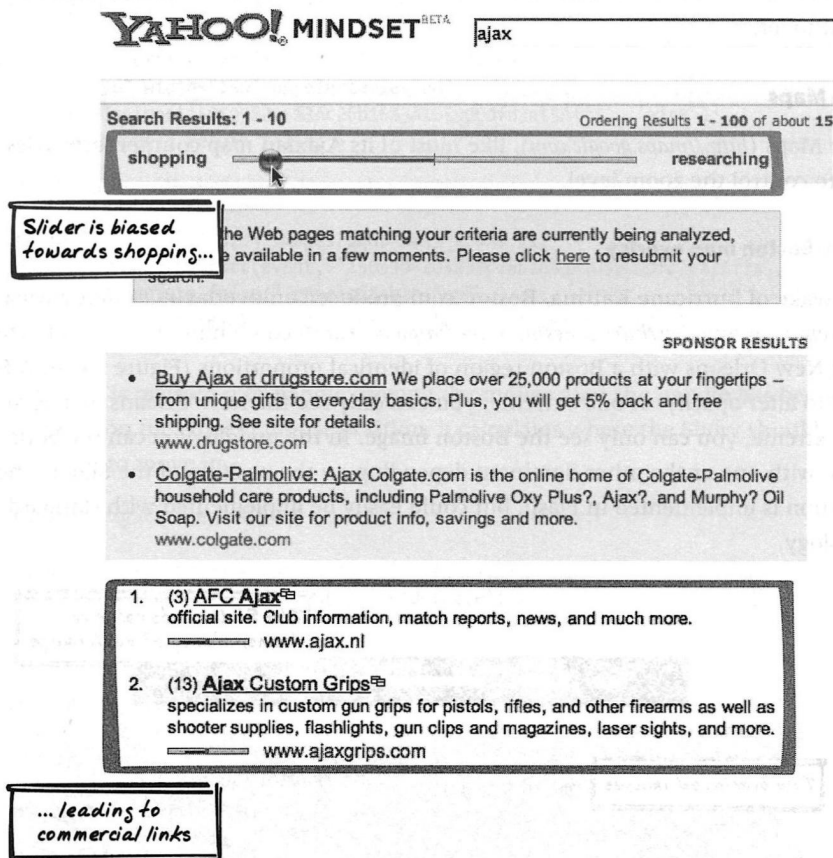


FIGURE 14-2. Yahoo! Mindset

Amazon Diamond Search

Amazon Diamond Search (<http://www.amazon.com/gp/gsl/search/finder/002-1527640-2908837?%5Fencoding=UTF8&productGroupID=loose%5Fdiamonds>) presents several elaborate Sliders (see Figure 14-5). Each lets you specify a range, which acts as a filter for the diamond search. There are several innovative aspects of the presentation, discussed next.

- The labels are dynamic. As you drag the handles, the labels show exactly the value that's been specified.
- There's a graphic inside the Slider, which represents the variable in each case. Price, for example, is shown as an increasing histogram, and Cut shows several diamonds of decreasing cut quality.
- The Slider graphic is faded outside the selection region.
- Each Slider has a *Microlink* (Chapter 15) that opens up an explanation about that variable.
- The Sliders are horizontal and parallel to each other. Unfortunately, the directions aren't aligned—price and carat increase from left to right, but cut quality increases from right to left.

Google Maps

Google Maps (<http://maps.google.com>), like most of its Ajaxian map counterparts, uses a Slider to control the zoom level.

Katrina-Boston map overlay

In the wake of hurricane Katrina, Boston.com produced an overlay map (http://www.boston.com/news/weather/hurricanes/multimedia/globe_map/) combining the flood-affected area of New Orleans with a Boston region of identical proportions (Figure 14-3). A Slider is used to alter opacity: at one extreme, you can only see the New Orleans image; at the other extreme, you can only see the Boston image. In the middle, you can see both images, with one or the other dominant depending on the position of the Slider. The application is implemented in Flash, but could easily be implemented with standard Ajax technology.

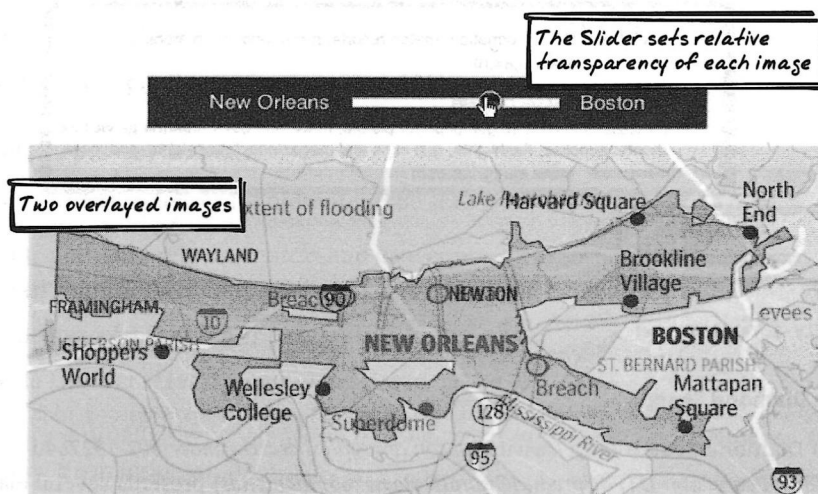


FIGURE 14-3. Katrina-Boston overlay map

WebFX Slider Demo

The WebFX Slider Demo (<http://webfx.eae.net/dhtml/slider/slider.html>) shows a few different Sliders, in both horizontal and vertical orientations. You can set the Slider directly and also update it with some text inputs.

Code Example: Yahoo! Mindset

The Yahoo! Mindset Slider (<http://mindset.research.yahoo.com/>) is created with several images. There are images for the vertical lines at the center and extremes. The main horizontal bar is created with a single vertical line; it's 1 pixel wide, but the image width is set to 150 pixels, thus creating a horizontal bar. There's a handler registered for the click event (which pulls the handle in that direction):

```

```

The Slider handle, called sliderball, is also an image:

```

```

A drag function is registered to handle moving the mouse after the handle has been selected. Based on the current mouse position, it calculates where the Slider should be and calls a function to move it:

```
function dragStart(e, sID, q) {
  ...
  document.onmousemove = function(e) { drag(e); };
  document.onmouseup = function(e) { endDrag(e); };
  ...
}

function drag(e) {
  ...
  var relativePos = e.clientX - sliderOffset;
  ...
  moveSlider(relativePos);
}
```

The moveSlider function redraws the handle based on its relative position (positive or negative offset from the center):

```
function moveSlider (relativePos) {

  var centerPoint = (maxRight - minLeft) / 2;
  var centerBuffer = 5;

  //the ball position is relative
  var ballPos = (-(maxRight - minLeft)) + (relativePos-(ballWidth/2));
  document.getElementById('sliderball').style.left = ballPos+'px';
```

```
...
}
```

Finally, when the Slider is released, the handle's position is finalized and an *XMLHttpRequest* Request Call is issued to bring the results in line with the new value:

```
function endDrag(e) {
    ...
    var relativePos = e.clientX - sliderOffset;
    drag(e);
    ...
    var sliderURI = "/searchify/slider?UserSession="+sessionID+
        "&SliderValue="+threshold+"&query="+query;
    (Sends XMLHttpRequest to SliderURI)
}
```

Alternatives

Conventional controls: radiobuttons and selector field

As mentioned in the solution, a Slider performs similar functionality to radiobuttons and selectors.

Related Patterns

Drag-And-Drop

The Slider handle is usually manipulated with a *Drag-And-Drop* (Chapter 15) action.

Metaphor

Sliders are a UI metaphor based on physical sliders in control devices such as audio-visual consoles.

Progress Indicator



Activity, Feedback, Hourglass, Meter, Progress, Waiting

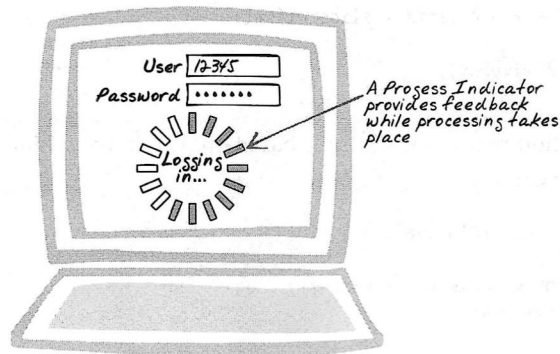


FIGURE 14-4. Progress Indicator