

Working with jQuery, Part 2: Intermediate JQuery: The UI project

Skill Level: Intermediate

[Michael Abernethy \(mike@abernethysoft.com\)](mailto:mike@abernethysoft.com)
Product Development Manager
Optimal Auctions

14 Apr 2009

The jQuery UI package aims to create a well-defined and reliable set of user interface widgets that you can reuse within your own Web applications. The goal is to provide well-tested widgets that go beyond those available in HTML `Input` elements, and ease the work of all user interface developers.

Introduction

Since the publication of this article, the 1.6 version of the UI project has been renamed 1.7.

The jQuery UI project is an exciting new branch of jQuery that promises to grow quickly in the next year. The UI library is expanding rapidly, much more rapidly than the jQuery core, as its developers push more features and fixes into each release. The funny thing about the UI package though, is that it's really a collection of user interface-related items, and can be broken down into 3 main modules: the widgets, which contain prebuilt and "skinnable" user interfaces that are ready to deploy to a Web site; the effects, which are very simple and straightforward things you can do to a page element (for example, shake it, explode it, and so on); and expanded mouse interaction with page elements (for example, dragging and dropping). The final aspect of the UI package is the ability to create your own "theme" for the prebuilt widgets, allowing you to make the widgets you download look like they were made especially for your Web site.

The history of the jQuery UI project is actually quite interesting and may help explain a little why the project is built the way it is, and why the three parts of the library

seem so different. The UI library actually started as several different plug-ins "back in the day" ("the day" meaning late 2007, which is decades ago in terms of jQuery). The plug-ins, which were all very popular and often downloaded, were created by the heavyweights of the jQuery community. It was decided that these popular plug-ins should be consolidated together into one larger plug-in and promoted as an official extension of the jQuery core code. This led to the release of UI version 1.5 in early 2008. However, after the release of this consolidated UI plug-in, developers who were trying to use it in their own code had comments (or complaints) that the code, while good, was all very different. After all, the plug-ins all had different authors, who all had different styles. A decision was made to try to adjust all of the plug-ins to have a common style. This resulted in all the 1.5.x releases, which you can see in the history of the UI library.

However, with the release of the 1.6 version of the UI library, the work of refactoring all of the code will be complete, and the promise of a UI library with a common style and common code will be met. One drawback with this release, though, is that the UI team made a decision to drop several widgets from its codebase in order to get the completed 1.6 version out the door. So, while v1.5 may have had three more widgets, v1.6 is supposedly faster and easier to work with. You can decide if the trade-off is worth it to you, but look for the dropped widgets to reappear in future releases of the UI library. (For the record, they were an autocomplete combobox, a magnifier, and a spinner).

Now that the history of the UI library is complete, you can start to look at all of the goodness that the library contains, and you can see why the members of the jQuery team are so excited about it. I, for one, am also excited about the potential of the library. For those of us old enough to remember working in Java™ UI, with AWT, (or those who have chosen to forget those days), remember the excitement you had when Swing was introduced, and a whole new world of widgets was introduced, one that made working with the user interface actually tolerable? I compare the *potential* of the UI library to Swing, in that it can add a whole new level of user interface to Web design, by adding more complex widgets to the arsenal of Web designers, while at the same time standardizing how they work and behave. So, while the number of widgets that are contained in the UI library may seem small now, here's hoping that one year from the release of v1.6, the UI library will have 10-20 new widgets that we, as developers, can use in our own Web applications.

Installing the UI

Downloading and installing the UI library isn't like installing the jQuery core or a plug-in. In an effort to minimize network traffic and connections, the UI team has created their own Web application where they ask you to prechoose the parts of the UI library you'd like to use and your desired compression scheme. Then they let you download the file directly. This has the benefit of allowing you to only provide the parts of the UI library that you intend to use, without sacrificing any performance of

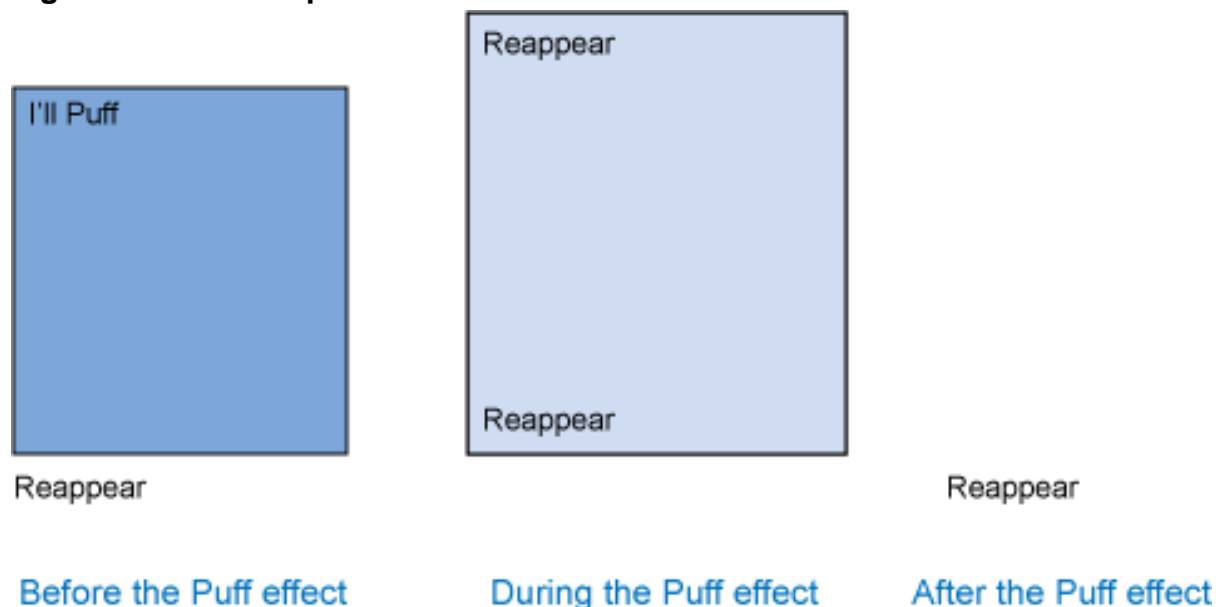
your application. However, this has the downside of requiring you to redownload the entire file if you ever decide to use a part you hadn't originally included in your custom-made file. Like all things in life, there's an upside and a downside to every decision.

Effects

The Effects module in the UI library contains "interesting" things you can do to page elements. I say interesting, because when you look at them as a whole, they are really quite a random hodgepodge collection of effects. It's like someone went through all the effects in Microsoft® PowerPoint and picked out a few of them to make into JavaScript effects. How these were chosen will probably never be known. How useful these will be on professional Web applications is still yet to be decided, as they look cool, but I question when and where some of them can be used.

The good news, though: they are extremely easy to use. The first set of effects help you hide/show elements on the page. They simply extend the jQuery library's built-in `hide()` and `show()` functions by letting you specify what type of effect you'd like it to use when it hides or becomes visible. So you can picture what I'm talking about, I'll attempt to illustrate the Puff effect with a flowchart in Figure 1, showing before, during, and after.

Figure 1. Puff example



Because it's difficult describing how animated effects look, you'll need to download the demo from this article to see them in action. However, I can show the code that goes into producing the static image in Figure 1. Your brain can take it from there.

Listing 1. Puff effect

```
$("#puffSample").click(function(){
    $(this).hide("puff");
});

$("#puffReappear").click(function(){
    $("#puffSample").show("puff");
});
```

As you can see, nothing too exciting to see here as far as code. It's rather straightforward how these show/hide effects work. The real challenge is figuring out a good time to use them in an appropriate manner, without causing your users to go "Oh wow! That was unnecessary."

Let's move on to the other effects contained in the Effects module. These are somewhat more practical, and I can imagine how they can be used in Web applications to improve the user experience (well, some of them at least). Again, it is difficult to show you the examples without actually being able to show them in action, so I recommend that you [download](#) the examples from this article and check them out for yourself.

The effects in this module are really a way to draw attention to certain elements on the page, and some of them can be used effectively in that regard. The effects are "bounce," which literally bounces an element as if it was dropped on some imaginary HTML trampoline; "highlight," which flashes the element with a yellow background (think Mac here); "pulsate," which makes the element toggle between visible and invisible; "scale," which shrinks the element by 50%; "shake," which shakes the element from right to left; and "size," which makes the element longer and flatter. They are also just as easy to use as the hide/show effects.

Listing 2. Bounce effect

```
$("#bounce").click(function(){
    $("#loginSample").effect("bounce");
});
```

This wraps up the Effects module of the UI library, and though it doesn't look that exciting on paper, there are really a few cool effects that you can work with if you find yourself in a situation where you might need an element exploding off of your page.

Interactions

Finally, I get to some things more exciting than making my DIV blow up on the screen. The Interactions provide a very powerful way of coding what can happen when you drag one element over another element, what can happen when you drop

one element on top of another element, and other exciting things you can do with two HTML elements.

Enabling interaction with dragging/dropping elements is an entirely new area of design for Web applications. Because most users are accustomed to static pages and static page elements, there are very few, if any, Web applications that allow some type of drag and drop interaction. Come to think of it, there are very few desktop applications that allow this type of unique interaction. They do exist, though, and I can think of one example specifically: Yahoo Fantasy Sports pages utilize the drag/drop model nearly perfectly, using them to let you shuffle your team rosters, rank your players before a draft, and move players to the cut list. Even if you are not into Fantasy Sports, I would encourage you to check out this application just to see the unique user interface design they've created there. I think it is one of the best I've seen for usage of the drag/drop model on the Web.

The Interactions module contains five types of interaction. The first three are:

- *Resizable* deals with making elements on a page resizable, allowing users to change their shape and adjusting all of the other elements on the page appropriately. This interaction would be very useful if you are creating a Web application with multiple portlets, allowing the user to custom design their own Web page. MyYahoo and iGoogle both let you customize your page, but they do it on their terms. You are limited to a certain number of columns with everything as they want it. Why not let the user *totally* define their pages?
- *Selectable* allows elements on the page to be grouped together, and then allows users to select a subset of those elements in the group. This type of interaction would be useful creating custom lists or selection widgets.
- *Sortable* interface. This allows your page elements to "snap" to positions in some sorted order. Imagine if you have a table of five rows. Utilizing the Sortable interaction, you could drag the fourth row to the second row, and the rest of the table would elegantly animate itself as a reaction. I encourage everyone to take a closer look at these three interactions, as they give developers some cool new ways to create widgets and Web app interaction.

I'd like to focus this section on the two interactions that I think are most exciting for the future of Web applications:

- Draggable
- Droppable

In the example I'm about to go through, I will be using both of them together, though this isn't required.

In this example Web application, I'm going to retool how people shop online. Instead of pressing a button to place something in the shopping cart, I want to replicate the "shopper's high" that many people get when they physically place an item in their shopping cart by utilizing the Draggable/Droppable interactions. Users will physically drag their items into the shopping cart. When an item drops into their cart, the number of items in the cart and the total price of all things in the cart will both automatically update.

Figure 2. Shopping cart example



Let's take a look at the surprisingly little amount of code it takes to accomplish this interaction. (The full working example is included in the sample code at the end of the article).

Listing 3. Shopping cart

```
// First, set up the shopping cart
// Give the cart an ID that we can reference in the jQuery code
// Give each span an ID so we can easily update the text in it

<p><span id=numItems>0</span> items in your cart.
<p>Your total is <span id=totalPrice>0</span>.

// Each product has code that looks like this. The entire product
// is wrapped in a DIV with a class of "product" and given a
// unique ID. It also has a span that defines the price of the object.
<p><div class=product id=ban> Bananas
```

```

<p>${<span class=price>1.99</span></div><br><br>
// define each DIV that has a "product" as "draggable"
// The draggable function has many options that define how the object
// will look and feel when dragged. There are many options, most not
// covered here, but I put a sampling in to whet your appetite.
$(".product").draggable({
  'opacity': 0.3, // make the object semi-transparent when dragged
  'revert': "valid", // snap the object back after it's been dropped
  'delay': 200, // delay 200 ms before starting to drag it
  'distance':4, // wait till it's been dragged 4 pixels before starting
  'helper':"clone" // keep the object where it is and use a helper to show dragging
});

// just like the draggable has many options, the droppable has many
// options as well. These options provide a variety of ways
// to offer reinforcement to the user about how the draggable/droppable
// relationship is to work
$("#cart").droppable({
  'accept':".product", // define which elements will trigger a 'drop'
  'activeClass':"border", // what class to add to the droppable while dragging
  'drop': function(e,ui){ // this function gets called when something is dropped
    // update the number of items in the cart
    $("#numItems").text(new Number($("#numItems").text()+1);
    var ID = $(ui.draggable).attr("id");
    // get the price from the object just placed in the cart
    var price = new Number($("#"+ID + ".price").text());
    // update the total price in the cart
    $("#totalPrice").text(new Number($("#totalPrice").text()+price);
  }
});

```

That's really all there is to this. When I created this demo I even surprised myself with how easy it was to create. Needless to say, with this being so easy and straightforward, creating a user interface that takes advantage of this type of interaction should become more prevalent. The other interactions mentioned in this section are also just as easy to work with, and offer their own unique aspects to make new interactions possible on your Web applications.

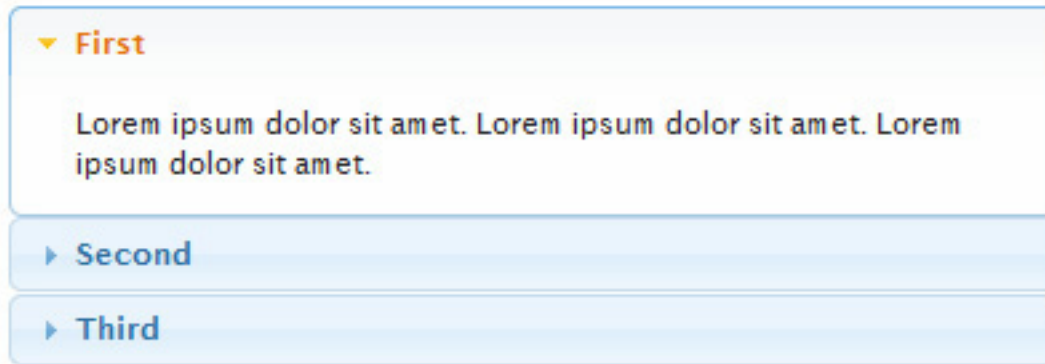
Widgets

Ahh, finally, a section of the UI code where I can actually show you images of what things look like. To me, the most exciting aspect of the UI library is the Widgets module, which contains premade, pretested widgets that entail aspects of Web applications that we developers run across all the time. By supplying these widgets, it saves us time, which of course we can spend surfing the Web and reading great articles like this one. I also suggest telling your boss that you slaved away on these widgets, so that he'll think you worked extra hard this year and give you that raise you've been striving for.

These screenshots are courtesy of the UI home page.

Accordian

Figure 3. Accordian



The accordion plug-in turns a list into a graphical representation of the list, allowing the users to move between each section of the list, viewing one, and only one, section at a time. In some UI circles, this widget is called the "Outlook Bar," after that program originated its usage a few years ago. One of the big mistakes with this widget is trying to get more than 1 section visible at the same time. This isn't that widget. This will display only one section at a time, and is effective for containing lots of information in a section, if the information is not dependent on another section in order to be effective.

This widget depends heavily on the proper setup of the HTML in order to work correctly. In other words, you need to set up your HTML in anticipation of using the accordion widget. This also takes into account the so-called "degrade gracefully" theory, in which users without JavaScript, or those using Internet Explorer 5, would still be able to view the page properly. In the example in Listing 4, note how the HTML is structured, as anyone wanting to use this widget will have to match it.

Listing 4. Accordion

```
// The accordion relies on the UL, LI, and A tags to properly configure itself
<ul id="accordionExample" class="ui-accordion-container" style="width:400px;">
  <li>
    <a href="#">Title 1</a>
    <div>Your text would go here</div>
  </li>

  <li>
    <a href="#">Title 2</a>
    <div>More text</div>
  </li>
</ul>

// yes, it's this simple
$("#accordionExample").accordion();
```

Slider

Figure 4. Slider



Listing 5. Slider

```
<div id="sliderExample" class="ui-slider-1">
<div class="ui-slider-handle"></div>
</div>

// this will set up the slider
$("#sliderExample").slider();

// this will set the maximum and minimum values of the slider
$("#sliderExample").slider({
  min: 10,
  max: 20
});

// this will get the value of the slider
$("#sliderExample").slider("value");
```

DatePicker

Figure 5. DatePicker



Listing 6. DatePicker

```
<input type="text" id="dateField">

// this will create a date picker, which pops up when the textfield gets focus
$("#dateField").datepicker();

// show the drop-down fields to let the user jump around months and years
$("#dateField").datepicker({
  changeMonth: true,
  changeYear: true
});

// show two months instead of one
$("#dateField").datepicker({
  numberOfMonths: 2
});

// because this widget has to support internationalization, it has a TON of
```

```
// functions that let you set every aspect of the calendar, to support any
// possible calendar settings. (Although it doesn't look like the jQuery team
// felt it was necessary to support the Mayan calendar. Guess 2012 is right
// around the corner.)
//
// This function lets you set the date format that will appear in the textfield.
$("#dateField").datepicker({
    dateFormat: 'dd/mm/yyyy'
});

// Gets the date back from the datepicker
var date = $("#dateField").datepicker("getDate");
```

Progress Bar

Figure 6. Progress Bar



Listing 7. ProgressBar

```
<div id="progressbarExample"></div>

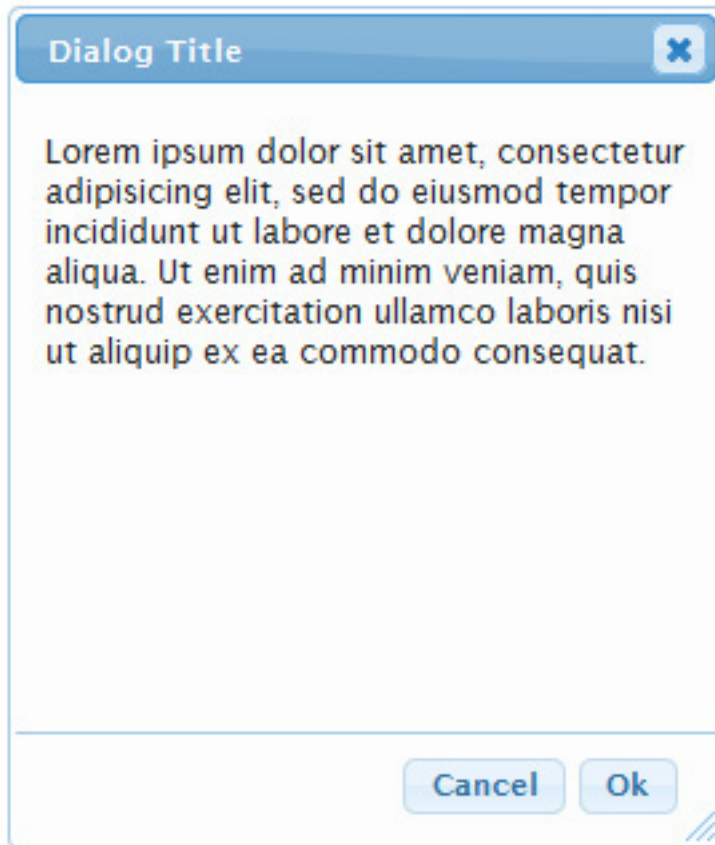
// make this div a progress bar
$("#progressbarExample").progressbar();

// start the progress bar
$("#progressbarExample").start();

// stop the progress bar
$("#progressbarExample").stop();
```

Dialog

Figure 7. Dialog



To those who are following this series, the Dialog widget may interfere with the usage you saw from the BlockUI plug-in from "[Working with jQuery, Part 1: Intermediate JQuery: Use plug-ins to create and extend the jQuery functions.](#)" Both of these will perform the desired dialog function. Which solution you use is up to you. The BlockUI gives you more flexibility about exactly how it will look, while this Dialog widget does much of the work for you.

Listing 8. ProgressBar

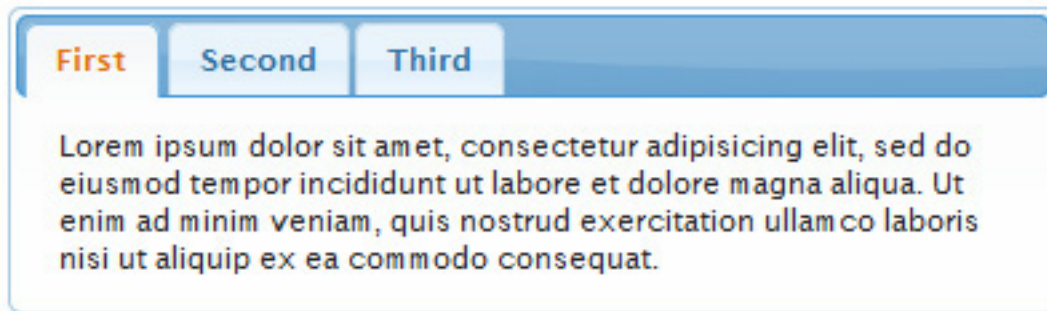
```
<div id="dialogExample" title="Example Dialog">Text here.  Warning!!</div>

// turn the DIV into a dialog
$("#dialogExample").dialog();

// open the dialog, then close the dialog
$("#dialogExample").dialog("open").dialog("close");

// make a dialog that the user can't drag around, is modal, and in the center of the page
$("#dialogExample").dialog({
  draggable: false,
  modal: true,
  position: "center"
});
```

Tabs

Figure 8. Tabs

Tabs is probably my least favorite of the widgets. I have never gotten it to work well, and have found that this particular widget relies very heavily on the CSS you provide to make it work correctly. The examples are lengthy and involved, and because of my existing bias against the widget, I'll choose not to review it here. Personally, I have found that creating a separate DIV yourself is almost as easy as working with this widget.

ThemeRoller

The final aspect of the UI library is called the "ThemeRoller." An odd name, but it is actually quite useful if you choose to use any of the widgets from the library in your code. As it stands now, the widgets have a default look and feel that I would guess does not match the look and feel that currently exists on your page. I would also guess that if you were to use these widgets you would want them to look like they belong on your site. So, the creators of the UI library have also taken the time to create, and greatly refine these past few months, a CSS generator that you can modify to match your own site, and then save the CSS file. The CSS generated will make all the widgets look and feel just like those on your site, ideally making the transition to using these widgets much easier on you.

One of the many benefits of the ThemeRoller is that it lets you create CSS for things that non-CSS experts (such as myself) couldn't normally do easily. For example, it lets you specify rounded corners on any part of the widget, and lets you create drop-down shadows as well. It has really taken the nerdy coding part of the CSS file out of the usage of the UI library, and allowed developers who aren't graphical artists to use and modify the widgets easily. Instead of hacking your way through a CSS file and coming out with a poor looking widget, the ThemeRoller lets you do it non-programmatically.

Finally, the ThemeRoller comes with many prebuilt themes that you can choose to have your widgets imitate. If you are like me, you can look at the prebuilt themes and realize that they are much, much nicer looking than anything that you've designed. Instead of placing the CSS from your site into the ThemeRoller, you can place the ThemeRoller CSS on your own site. Consider this the "cheap" way to get a

professional looking Web application for free!

Conclusion

This brings to a close the article on the UI library. The UI library is a quickly evolving project that is constantly reformatting and improving its offerings. In the last year, it has come a long way from its beginning as a thrown-together assortment of popular plug-ins. I hope to see this rapid pace of growth continue because I'm excited about the possibilities that the library can offer developers. While I may have been negative on the Effects module, the Interaction module and the Widget module offer many compelling reasons to give the UI library a chance in your Web applications.

In this article I went through all of the widgets that are offered in the 1.6 version of UI. These 6 widgets are the first of hopefully many widgets you'll see from the UI team. I'm hopeful that in the future you'll see such common desktop widgets as a colorpicker, a time picker, an autocomplete drop-down, a menubar, a toolbar, and so on. As I've stressed throughout my many articles, one of the overarching goals of the jQuery team is to make creating a Web application as easy as creating a desktop application. I also think it's important for users that their Web apps look as close to their desktop apps as possible, as this can only help build acceptance and regular use of them.

Downloads

Description	Name	Size	Download method
Examples.zip	examples.zip	126KB	HTTP

[Information about download methods](#)

Resources

- Download the [1.3.2 Minimized jQuery](#), which was the latest stable version at the time of this writing, and drop it in your own code.
- Read the complete [jQuery API page](#) to see all of the available functions in the library.
- Download the [jQuery UI Library](#) to get all the widgets and effects discussed here.
- View the [jQuery UI Demo page](#) to see all widgets in action.
- Visit the [ThemeRoller page](#) to create your own look and feel for the widgets.
- Get a thorough and complete background on CSS, JavaScript, and any other Web language at [W3Schools](#).
- Read the first 3 articles in my jQuery series, that deal with an introduction to the library. "[Working with jQuery, Part 1](#)," "[Working with jQuery, Part 2](#)," "[Working with jQuery, Part 3](#)."
- "[Ajax overhaul, Part 2: Retrofit existing sites with jQuery, Ajax, tooltips, and lightboxes](#)" (developerWorks, May 2008) gives you a look at some of the other plug-ins available for jQuery.

About the author

Michael Abernethy

In his 10 years in the technology field, Michael Abernethy has worked with a wide variety of technologies and clients. He currently works as the product development manager for Optimal Auctions, an auction software company. His focus is on Rich Internet Applications and making them both more complex and simpler at the same time. When he's not working at his computer, he can be found on the beach in Mexico with a good book.

Trademarks

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.