

Incorporating MooTools into your Rails Apps

Robert Dempsey

[Atlantic Dominion Solutions, LLC](#)

May 14, 2007

I wrote a little blurb a while ago about [MooTools](#) not playing well with [Prototype](#) (at least not the latest BETA version – we love the new stuff). This continues to be the case as there appears to be conflicts with the two libraries. So for now, we have to choose between Prototype/[Scriptaculous](#) or MooTools.

I am not one to give up without a fight, and I really wanted to use MooTools as it has some killer effects. This past Friday I sat down and figured out how to use it, and am happy to report that it is super easy. In fact, it took me about 5-10 minutes to recreate the main navigation menu on the [Rails For All](#) site using the [Accordion](#) plugin that MooTools provides. This tutorial will give a quick rundown of how to use MooTools in your Rails app.

I started the day by watching the [Beauty in Design](#) video tutorial, which I highly recommend. After that, I created a new Rails app, downloaded the newest version of MooTools, and included the mootools.js file in head tag of my application layout.

```
<%= javascript_include_tag 'mootools' %>
```

The next step was to create my navigation menu. For the sake of brevity, I will include two levels of the menu here. On the Rails For All site we have multiple language translations so we have the menu in a partial, rendering the language based on a session variable. Here's a chunk of code from the menu partial.

```
<ul id="nav">
  <li class="box_title"><a href="#">Select your language</a>
    <ul id="nav2" class="expand">
      <li><%= link_to "English", {:controller => 'info', :action => 'set_locale', :locale => 'en', :id =>
params[:id]} %></li>
      <li><%= link_to "Spanish", {:controller => 'info', :action => 'set_locale', :locale => 'es', :id =>
params[:id]} %></li>
    </ul>
  </li>
  <li class="box_title"><a href="#">Rails For All</a>
    <ul id="nav2" class="expand">
      <li><%= link_to "Home", home_path, :title => "Rails For All" %></li>
      <li><%= link_to "About", about_path, :title => "About Rails For All" %></li>
      <li><%= link_to "Mission and Principles", mission_and_principles_path %></li>
      <li><%= link_to "Sponsorship", sponsorship_path, :title => "Sponsors of Rails For All" %></li>
      <li><%= link_to "Contact Us", contact_us_path, :title => "Contact Rails For All" %></li>
    </ul>
  </li>
</ul>
```

The list items in the top level list are effectively our menu headings, and will be the links that active the accordion effect and expand the menu. The top level list items then contain a list of items that are the “sub-menu” items that appears when the menu is expanded.

Now, to make the Accordion effect work, we need a little JavaScript magic. Adding the following script

to our head tag turns our menu into the cool accordion effect we want.

```
<script type="text/javascript">
  window.onload = function() {
    var titles = document.getElementsByClassName('box_title');
    var expanders = document.getElementsByClassName('expand');
    // Create the accordion
    var myEffect = new Fx.Accordion(titles, expanders, {show: 0});
  }
</script>
```

We put this script into the head tags to ensure that the effect is applied when the pages are loaded. Let's take a look at what we have here.

```
window.onload = function() {}
```

The above creates a new function that will fire off when the current window is loaded.

The Accordion effect takes three arguments: togglers, elements, and options. Toggles are the clickable elements that activate the Accordion effect (our top-level menu items); elements are the page elements that will have the effect applied to them (our sub-menu items); options are available options for the effect. Our one liner below creates an array of elements called title, that holds the page elements that have a class of “box_title.”

```
var titles = document.getElementsByClassName('box_title');
```

The next line of script creates an array of elements that have a class of “expand.”

```
var expanders = document.getElementsByClassName('expand');
```

To wrap it up, we add the following line to create the MooToolsAccordion effect and make the magic happen.

```
var myEffect = new Fx.Accordion(titles, expanders, {show: 0});
```

The one option argument that we have added is show. We tell the effect to show the first item in the array of togglers, which automatically expands our first menu item.

That's it! The result is a menu that works swiftly, allows us to have a more compact page with a great many menu options, and not overwhelm our users with too many menu options at once.

We definitely recommend MooTools. It has a small footprint, loads quickly, isn't choppy in its application of effects, and is super easy to use. [Download the latest](#) and play around with it. Enjoy!