/jQuery/ building dynamic forms

Your essential CD
All the files you'll require for this tutorial can be found on this issue's CD.

Part one **Chris Coyier** from CSS-Tricks walks us through building a dynamic form. Here he puts solid HTML and CSS in place before using jQuery to add dynamic functionality in part two

Knowledge needed A basic working knowledge of HTML and CSS is ideal, but not required.

Requires Your code editor of choice

Project time The source code for this project is provided, so you can open up the example and get started right away.

Forms largely power the interactivity of the web, but most forms lack any dynamic features that could make them less intimidating, more friendly and more fun! Using the powerful (but easy to learn) jQuery JavaScript library, we can watch for events on our form inputs and dynamically change the form to suit our needs.

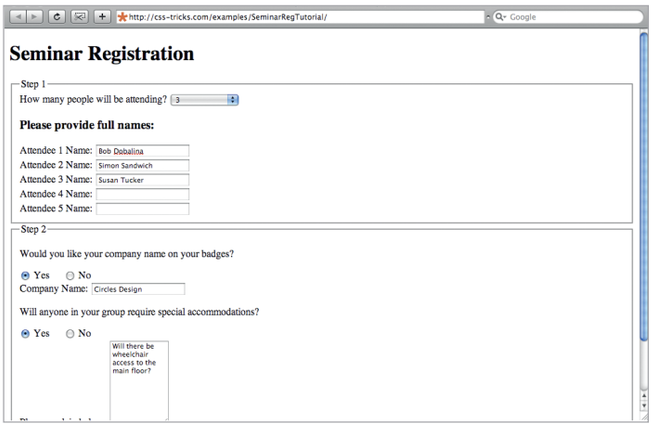
Instead of being abstract about these ideas, let's actually walk through building a dynamic form from start to finish. We'll be building a totally fictitious registration form for a totally generic seminar. The idea is to demonstrate the concepts, not to get anyone registered for a seminar, so forgive me if the example feels somewhat incomplete for a real seminar registration form. In the same way that a picture is worth a thousand words, a live example is worth a thousand written tutorials. The complete demo for this tutorial is available on this issue's CD, as well as live on the web at css-tricks.com/examples/SeminarRegTutorial/.

Basic planning

Let's get our goals straight before we dive into any code. The idea is seminar registration, but from a company perspective.

One person will be filling out the form, but they may be registering multiple people. We'll need to capture the names of each attendee, and this will be required before the form can be submitted.

Additionally we'd like to ask some generic questions about the group as a whole. Then we're going to require a little enthusiasm (in the form of a checkbox) before form submission.



Step by step The questions on our form are grouped into three steps to help our users along; HTML provides the perfect element for achieving this in the **<fieldset>**

This breaks nicely into three steps, which we can use in our design to help lead users along. Let's break our questions into Step 1, Step 2 and Step 3. Step 1 is to find out how many people will be attending. The plan here is to show the question and a drop-down menu which defaults to 'Please Choose'. The drop-down will have the options of 1-5. When the number is chosen, inputs are revealed which ask for each of their names. This way, the user is only presented with input fields they actually need to fill out. We avoid visual clutter and the off-putting feeling of seeing loads of empty input fields.

> One of our considerations is that we don't forget the less technologically enabled

Step 2 involves two simple yes/no questions. The first is: would you like your company name on your badges? If the answer is yes, the user will be presented a field to supply that name.

The second is: will anyone in your group require special accommodation? If yes, the user will be presented a text area to explain. This will need to be larger to accommodate longer answers.

For Step 3, just for fun, we'll ask our registrants: are you ready to rock?. The 'complete registration' button will be inactive until this checkbox is checked. This could just as easily be something like 'I agree to the Terms & Conditions'. Additionally, the 'complete registration' button requires all questions from step 1 and step 2 to be complete before activating.

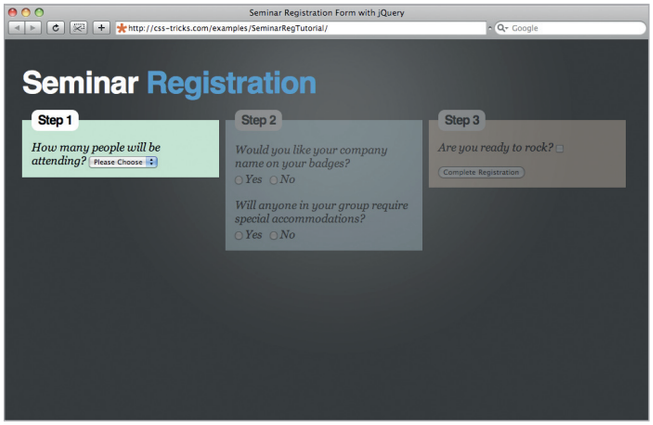
HTML markup

One of our most important considerations is that we don't forget our less technologically enabled brethren as we move forward with our fancy scripting. At the time of this writing, the NoScript plug-in for Firefox (which forcibly prohibits JavaScript from running in Firefox) has just over 30million downloads. We need to make sure that our HTML markup, CSS and JavaScript are all written so that the form absolutely works under any circumstances.

HTML provides the perfect element for breaking our form into steps, the <fieldset>. Let's take a look at the HTML for each step:

For Step 1:

```
<fieldset id="step_1">
  <legend>Step 1</legend>
  <label for="num_attendees">
    How many people will be attending?
  </label>
  <select id="num_attendees">
    <option id="opt_0" value="0">Please Choose</option>
    <option id="opt_1" value="1">1</option>
    <option id="opt_2" value="2">2</option>
    <option id="opt_3" value="3">3</option>
```



First hurdle Step 1 asks how many people will be attending. Our drop-down menu for this is made up of the **<select>** element and its child **<option>** elements

```
    <option id="opt_4" value="4">4</option>
    <option id="opt_5" value="5">5</option>
  </select>
  <br />
  <div id="attendee_1_wrap" class="name_wrap push">
    <h3>Please provide full names:</h3>
    <label for="name_attendee_1">
        Attendee 1 Name:
    </label>
    <input type="text" id="name_attendee_1" class="name_input"></input>
  </div>
  <div id="attendee_2_wrap" class="name_wrap">
    <label for="name_attendee_2">
        Attendee 2 Name:
    </label>
    <input type="text" id="name_attendee_2" class="name_input"></input>
  </div>
  <!-- wraps 3-5 follow the exact same for as the "attendee_2_wrap" -->
</fieldset>
```

The **<legend>** element is the semantically correct element to use to title a fieldset. Typically, browsers position the legend vertically centred along the top border of the fieldset, which is kind of cool and something we'll exploit for our design. There are some quirks to this, though, which we'll have to fight with an IE-specific style sheet.

The meat of Step 1 is our drop-down menu, which is the **<select>** element and its child **<option>** elements. But we'll need to ask our question first, which takes the form of a **<label>** element.

After the **<select>** element, we have a series of five divs, each of which has a class of **name_wrap** as well as a unique ID. Each of those divs contains an input and a label for one attendee name.

Having a wrapping div is the easiest way to handle hiding and showing these inputs, as we'll only need to target the wrap instead of both the input and the label. Giving the wrap both a generic class and a unique ID is useful for both our CSS and JavaScript, as sometimes it will be appropriate to target them all as a group, and sometimes it will be appropriate to target individuals. Notice the first one contains a header element asking the user to provide full names. This is just a convenient place to tuck it, since the first wrap will always be shown when any number of attendees is chosen.

For Step 2:

```
<fieldset id="step_2">
  <legend>Step 2</legend>
  <p>
    Would you like your company name on your badges?
  </p>
  <input type="radio" id="company_name_toggle_on"
name="company_name_toggle_group"></input>
  <label for="company_name_toggle_on">Yes</label>
   
  <input type="radio" id="company_name_toggle_off" name="company_
name_toggle_group"></input>
  <label for="company_name_toggle_off">No</label>
  <div id="company_name_wrap">
    <label for="company_name">
      Company Name:
    </label>
    <input type="text" id="company_name"></input>
```

Fixing Internet Explorer

There are two significant design problems when viewing our page in Internet Explorer. Both of are present in version 6, 7, and the most beta (2) of version 8, so we'll use conditional commenting to target IE as a whole and apply our fixes. The HTML inside of conditional comments is only seen by the targeted browser(s), so we can link to style sheets, JavaScript, even display content exclusively for Internet Explorer. Learn more about conditional style sheets and how to use them at css-tricks.com/how-to-create-an-ie-only-stylesheet/.

Legend/fieldset positioning

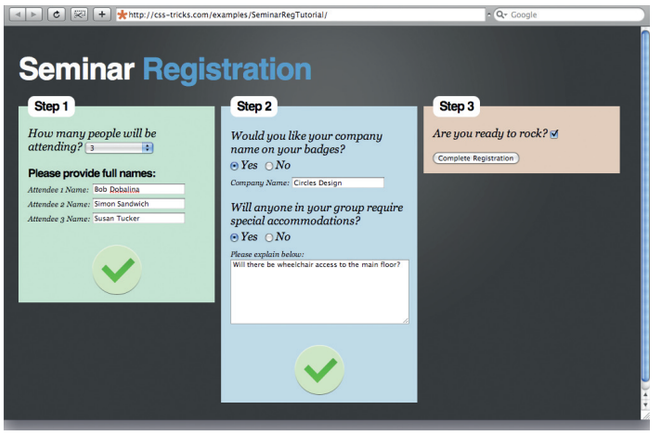
Fieldsets and their legends are some of the most hilariously buggy elements when dealing with Internet Explorer. Our problem is that IE pushes our legend box inside the fieldset because of the padding on the fieldset. One solution that may pop to mind right away is to use some negative top margin on the legend to pull it back up. This would work well, except for that fact that we're using transparency on the fieldsets, which (for some reason) hides all overflow and cuts off any of our legend hanging above. The other solution would be to set a relative position on it and kick it up with a negative top value. This solves the cut-off problem, but introduces a new one. Now any transparency applied to the fieldset is ignored on the legend. Headache-inducing, but this one we can fight. Within our conditional comments, we'll apply special CSS and JavaScript to deal with it:

```
<!--[if IE]>
  <style type="text/css">
    legend {
      position: relative;
      top: -30px;
    }
    fieldset {
      margin: 30px 10px 0 0;
    }
  </style>
  <script type="text/javascript">
    $(function(){
      $("#step_2 legend").css({ opacity: 0.5 });
      $("#step_3 legend").css({ opacity: 0.5 });
    });
  </script>
<![endif]-->
```

Rounded corners

The other major design difference is the rounded corners on those legends. I'm usually more than happy to use the vendor-specific CSS methods for rounded corners and let IE be content with hard corners. That's what we call progressive enhancement 'round here. But if rounded corners are vital to the design, there are ways to bring them into the party even in IE. Since we're using jQuery, the easiest way would be to use a jQuery plug-in to handle it. Simply include the plug-in file and activation of it within the conditional comments so only IE loads and uses it. See a demo and download the jQuery plug-in here (www.methvin.com/jquery/jq-corner-demo.html).

>>



Final furlong Step 3 contains the final submit button; we have to disable this until the form is filled out. This is done in XHTML by including **disabled="disabled"** as an attribute

```
>>    </div>
      <p>
    Will anyone in your group require special accommodations?
  </p>
  <input type="radio" id="special_accommodations_toggle_on"
name="special_accommodations_toggle"></input>
  <label for="special_accommodations_toggle_on">Yes</label>
   
  <input type="radio" id="special_accommodations_toggle_off"
name="special_accommodations_toggle"></input>
  <label for="special_accommodations_toggle_off">No</label>
  <div id="special_accommodations_wrap">
    <label for="special_accomodations_text">
      Please explain below:
    </label>
    <textarea rows="10" cols="10" id="special_accomdations_text"></
textarea>
  </div>
</fieldset>
```

Here we ask our two questions that have to do with the group as a whole. Should 'Yes' be chosen for either, our plan is to reveal an input area at that time. Otherwise, those input areas will be hidden. So we wrap those areas in divs with the IDs **company_name_wrap** and **special_accommodations_wrap** respectively.

Notice that the questions themselves are in paragraph (**<p>**) tags this time instead of labels. This can feel a little weird, but in the case of radio buttons, 'Yes and 'No' are the labels, not the question itself.

The only other piece of note is the ** ** character, which represents an 'em space'. Because our radio buttons and labels are both intentionally inline elements, this just allows us to push the 'No' option a bit over to the right, visually separating the options.

For Step 3:

```
<fieldset id="step_3">
  <legend>Step 3</legend>
  <label for="rock">
    Are you ready to rock?
  </label>
  <input type="checkbox" id="rock"></input>
  <input type="submit" id="submit_button"  value="Complete Registration"></
input>
</fieldset>
```

Fairly straightforward and simple here in Step 3. One thing to note is that this step contains the final submit button. While we intend to disable this button until the form is filled out, the HTML doesn't include any such disabling.

Disabling of inputs is done in XHMTL by including **disabled="disabled"** as an attribute. There is no such thing as disabled="false", or anything else which would re-enable the input. If an input has the disabled attribute at all, it's disabled. So, we'll be applying and removing this attribute from the input, all from the JavaScript.

There is some CSS that I'm not showing here, such as the reset, basic page layout, and typography. Refer to the code samples on the CD for the complete file. Next, we'll look at fieldsets and legends:

```
fieldset {
  width: 280px;
  padding: 15px;
  float: left;
  border: none;
  margin: 0 10px 0 0;
  }
  fieldset#step_1   {
  background: #b2e7ca;
  }
  fieldset#step_2   {
    background: #b2d9e7;
  }
  fieldset#step_3 {
    background: #e7c7b2;
  }
legend {
  font-weight: bold;
  font-size: 20px;
  background: white;
  -moz-border-radius: 10px;
  -webkit-border-radius: 10px;
  padding: 5px 10px;
  letter-spacing: -1px;
}
```

Fieldsets will be the 'boxes' that we use to house each of our three steps. While being the semantically correct element to use, they aren't without their quirks. For example, Firefox won't obey a min-height on a fieldset. In Internet Explorer, the padding on the fieldset affects the positioning of the legend element. Of note here is how we set all the common attributes of the fieldsets generically, while using the IDs of the fieldsets to change only the background colour. Efficiency is our friend!

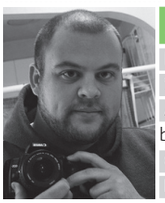
Now we'll consider other form elements:

```
option {
  padding: 0 5px;
}
#special_accommodations_wrap textarea {
  width: 100%; height: 100px;
}
```

I like to apply a little padding back to the **<option>** elements, which are the ones that make up the drop-down menu. Without it, the text uncomfortably butts up against the edge of the drop-down.

In XHTML, textareas are required to contain both the 'rows' and 'cols' attributes, although using these attributes to do any accurate sizing is no good. Thankfully, they're easy to override by setting width and height in the CSS.

Our solid HTML is in place, ensuring functionality all by itself, and our CSS is beautifying our design. In part two in issue 187, we'll get into the dynamic functionality of this form using jQuery. ●



About the author

Name Chris Coyier
Site CSS-Tricks (css-tricks.com)
Areas of expertise Design, user experience and standards-based XHTML/CSS
Company Chatman Design (chatmandesign.com)
How do you like your eggs? Scrambled with ketchup