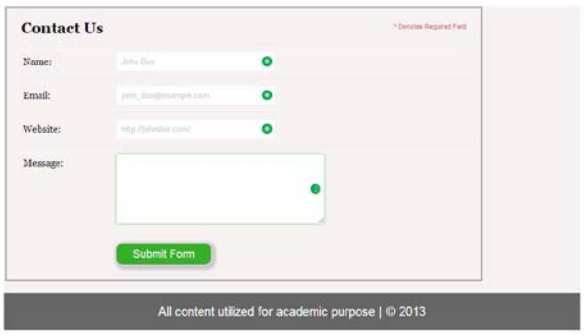
Form Elements in HTML5 - Part One

Part One:

There are a number of new HTML5 form elements, and almost all of them are mostly backward compliant and require little effort to incorporate into your web site. In this walkthrough I will cover the basics of simple form development and review the new HTML5 attributes *placeholder, focus, autocomplete, required, type, valid and pattern.*

The form we are going to build will be on an administration page for the user story "the owner would like to be able receive emails form users". I will not be adding a post URL to send the data to a back end that is covered in a different lesson. Our form will contain the objects from the screenshot bellow.



In this walkthrough will demonstrate how to create and style the following aspects of our HTML5 form in three parts.

Part One	• The basic html form setup • Adding the placeholder attribute • Understanding the autocomplete attribute • Understanding an elements focus state • Adding the basic CSS styling of the form • Styling the header area of the form
Part Two	• Styling the form submit button • Styling the core form input elements • Adding the "required" attribute in HTML5 • Styling with the pseudo class selector: required • Understanding the HTML5 "type" attribute

Part Three

- Adding the pseudo classes :valid and :invalid
- Understanding the “pattern” attribute
- Hiding the form field hints with display: none
- Using the “+” adjacent css selector

The Basic Html Form Setup

Step One-Start with a form:

```
<form class="contact-form" action="#" method="post" name="contact-form1">
</form>
```

To keep our form content organized and structured we will first enclose the form objects in a fieldset, then each of our form elements (label, input, etc.) will be in an unordered list. So let's start by creating the form heading and our first input element:

```
<fieldset>
<ul>
    <li>
      <h2>Contact Us</h2>
      <span class="required-notification">* Denotes Required Field </span>
    </li>
    <li>
        <label for="name"> Name: </label>
        <input type="text" id="name" />
    </li>
      <li>
         <label for="email"> Email: </label>
         <input type="email" id="email" />
         <span class="form-hint">Proper format "name@something.com"</span>
      </li>
      <li>
         <label for="website"> Website: </label>
         <input type="website" id="website" />
         <span class="form-hint">Proper format
           "http://someaddress.com"</span>
      </li>
      <li>
         <label for="message">Message:</label>
         <textarea id="message" cols="40" rows="6" > </textarea>
      </li>
      <li>
         < input type="submit">Submit Form</input>
      </li>
</ul>
</fieldset>
```

Adding the Placeholder Attribute

One of the first improvements HTML5 brings to web forms (one you're probably already familiar with) is the ability to set the placeholder text. Placeholder text is displayed when the input field is either empty or not in focus. Let's add the placeholder attribute to our input elements (name, email and website.). This will help the user understand what they should enter in each field.

- `placeholder="John Doe"`
- `placeholder="john_doe@example.com`
- `placeholder="http://johndoe.com/"`

It is a good idea to style your placeholder text in your CSS, that will help a user understand to change these fields, there are some browser prefixes to help you. Note: Check to see how they work in versions of Internet Explorer.

```
 :-moz-placeholder { color: #cccccc; }

::-webkit-input-placeholder { color: #cccccc; }
```

Understanding the Autocomplete Attribute

The autocomplete attribute helps users complete forms based on earlier input. The attribute has been around since IE5.5 but has finally been standardized as part of HTML5. The default state is set to on. This means that generally we won’t have to use it. However, if you want to insist that a form field be entered each time a form is completed (as opposed to the browser auto filling the field), you would implement it like so:

```
<input type="text" name="tracking-code" id="tracking-code"
autocomplete="off">
```

The autocomplete state on a field overrides any autocomplete state set on the containing form element.

Understanding an Elements Focus State

When a text field is active it is said to be focused. “Setting the focus means making the field active so the cursor is blinking and ready to accept text input.” The default of most web page forms is to make the first text field or text area automatically active when the page loads. “This means that as soon as the page has loaded the user can begin typing without having to click in the text field first.” The User Agent style sheet automatically adds some styling to input elements when they are in focus. If you require it you can override these defaults with: **`form: focus {outline: none;}`** However – please remember **Jakob's Law of the Web User Experience** “Users spend most of their time on other sites”. *We will modify the focus state latter in this walk-through when style the submit button.*

Adding the Basic CSS Styling of the Form

Let add some basic styling to our form and style our list elements to give our form some structure:

```
fieldset {width: 750px;
       border: 2px solid #999;
       }

.contact-form h2, .contact-form label {
      font-family: Georgia, Times, "Times New Roman", serif;
      }

.form-hint, .required-notification { font-size: 11px; }
```

```
.contact-form ul {
     width:750px;
     list-style-type: none;
     list-style-position: outside;
     margin: 0px;
     padding: 0px;
}

.contact-form li{
     padding: 12px;
     border-bottom: 1px solid #eee;
     position: relative;  }
```

Also, let's add a slight border to the top section of the form to separate the heading from the controls. We can accomplish this by using the :first-child selector. This will select, as the name implies, the first element in the <ul> list.

This adds some useful visual sectioning to our form. Keep in mind that these CSS selectors are not supported in older browsers. Since this is not vital to key functionality, we're rewarding those users who use current browsers.

```
contact-form li:first-child {
      border-bottom: 1px solid #777;
     }
```

Styling the Header Area of the Form

Let's style the header section of our form. This includes the heading tag and the notification that informs users that the asterisk symbol (*) indicates the required fields.

```
.contact-form h2 {
     margin: 0;
     display: inline;
     }

.required-notification {
     color: #d45252;
     margin: 5px 0 0 0;
     display: inline;
     float: right;
     }
```

End of Part One

Save and text your code then upload a copy to the web server. At this point we will pause before continuing on, in the second part of this lesson we will:

- Styling the form submit button
- Styling the core form input elements
- Adding the "required" attribute in HTML5
- Styling with the pseudo class selector: required
- Understanding the HTML5 "type" attribute

Note: If you have a web enabled mobile device, bring it to the next class to demonstrate one of the new HTML5 features.

References:

Clark, R. (2013, February 26). Html5 forms introduction and new attributes. Retrieved from http://html5doctor.com/html5-forms-introduction-and-new-attributes/

Contributing Authors. (Unknown). Focus a textarea or text field. Retrieved from http://www.mediacollege.com/internet/javascript/form/focus.html

Tupence. (2012). Grains of sand -css3 demos - button styling . Retrieved from http://cssdemos.tupence.co.uk/button-styling.htm

Nielsen, J. (2013, January 02). Bring your forms up to date with css3 and html5 validation. Retrieved from http://webdesign.tutsplus.com/tutorials/bring-your-forms-up-to-date-with-css3-and-html5-validation--webdesign-4738

The W3C Consortium. (2012). Css outlines. Retrieved from http://www.w3schools.com/css/css_outline.asp