**New Input Attributes:**

Autocomplete

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.

Example:

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

Autofocus

autofocus does exactly what it says on the tin. Adding it to an input automatically focuses that field when the page is rendered. As withplaceholder, autofocus is something that we used JavaScript for in the past.

Traditional JavaScript methods do, though, have some serious usability problems. For example, if a user begins completing the form before the script loads, they will (jarringly) be returned to the first form field when the script is loaded. The autofocus attribute in HTML5 gets around this issue by focusing as soon as the document loads, without having to wait for the JavaScript to be loaded. However, we only recommend using it for pages whose sole purpose is the form (like Google) to prevent the usability issues.

Example:

<input type="text" name="first-name" id="first-name"autofocus>

Form

The form attribute is used to associate an input, select, ortextarea element with a form (known as its form owner). Usingform means that the element doesn’t need to be a child of the associated form and can be moved away from it in the source. The primary use case for this is that input buttons that are placed within tables can now be associated with a form.

Example:

<input type="button" name="sort-l-h" form="sort">

**formaction**

formaction specifies the file or application that will submit the form. It has the same effect as the action attribute on the form element and can only be used with a submit or image button (type="submit" ortype="image"). When the form is submitted, the browser first checks for a formaction attribute; if that isn’t present, it proceeds to look for an action attribute on the form.

<input type="submit" value="Submit"formaction="process.php">

**formenctype**

formenctype details how the form data is encoded with the POST method type. It has the same effect as the enctype attribute on the form element and can only be used with a submit or image button (type="submit" or type="image"). The default value if not included is application/x-www-formurlencoded.

<input type="submit" value="Submit"formenctype="application/x-www-form-urlencoded">

**formmethod**

formmethod specifies which HTTP method (GET, POST, PUT, DELETE) will be used to submit the form data. It has the same effect as the method attribute on the form element and can only be used with a submit or image button (type="submit" or type="image").

Example:

<input type="submit" value="Submit" formmethod="POST">

**formtarget**

formtarget specifies the target window for the form results. It has the same effect as the target attribute on the form element and can only be used with a submit or image button (type="submit" ortype="image").

Example:

<input type="submit" value="Submit" formtarget="\_self">

### novalidate and formnovalidate

The novalidate and formnovalidate attributes indicate that the form shouldn’t be validated when submitted. They are both Boolean attributes. formnovalidate can be applied to submit or image input types. The novalidate attribute can be set only on the form element.

An example use case for the formnovalidate attribute could be on a “save draft” button, where the form has fields that are required for submitting the draft but aren’t required for saving the draft.novalidate would be used in cases where you don’t want to validate the form but do want to take advantage of the more useful user interface enhancements that the new input types offer.

Example:

<form action="process.php">  
  <label for="email">Email:</label>  
  <input type="text" name="email"value="gordo@example.com">  
  <input type="submit" formnovalidate value="Submit">  
</form>

And this example shows how to use novalidate:

<form action="process.php" novalidate>  
  <label for="email">Email:</label>  
  <input type="text" name="email"value="gordo@example.com">  
  <input type="submit" value="Submit">  
</form>

List

The list attribute enables the user to associate a list of options with a particular field. The value of the list attribute must be the same as the ID of a datalist element that resides in the same document. Thedatalist element is new in HTML5 and represents a predefined list of options for form controls. It works in a similar way to the in-browser search boxes that autocomplete as you type.

Example:

<label>Your favorite fruit:  
<datalist id="fruits">  
  <option value="Blackberry">Blackberry</option>  
  <option value="Blackcurrant">Blackcurrant</option>  
  <option value="Blueberry">Blueberry</option>  
  <!-- … -->  
</datalist>  
If other, please specify:  
  <input type="text" name="fruit" list="fruits">  
</label>

### multiple

We can take our lists and datalists one step further by applying the Boolean attribute multiple to allow more than one value to be entered from the datalist.

Example:

<label>Your favorite fruit:  
<datalist id="fruits">  
  <select name="fruits">  
    <option value="Blackberry">Blackberry</option>  
    <option value="Blackcurrant">Blackcurrant</option>  
    <option value="Blueberry">Blueberry</option>  
    <!-- … -->  
  </select>  
If other, please specify:  
</datalist>  
  <input type="text" name="fruit" list="fruits"multiple>  
</label>

multiple isn’t exclusively for use with datalists, though. A further example for multiple might be for email addresses when sending items to friend or the attachment of files, as shown here:

<label>Upload files:  
<input type="file" multiple name="upload"></label>

Pattern

The pattern attribute is likely to get a lot of developers very excited (well, as excited as you can get about form attributes). It specifies a JavaScript regular expression for the field’s value to be checked against. pattern makes it easy for us to implement specific validation for product codes, invoice numbers, and so on. The possibilities forpattern are wide-ranging

Example:

<label>Product Number:  
<input pattern="[0-9][A-Z]{3}" name="product"type="text" title="Single digit followed by three uppercase letters."/>  
</label>

### placeholder

First up is the placeholder attribute, which allows us to set placeholder text as we would currently do in HTML4 with the valueattribute. It should only be used for short descriptions. For anything longer, use the title attribute. The difference from HTML4 is that the text is only displayed when the field is empty and hasn’t received focus. Once the field receives focus (e.g., you click or tab to the field), and you begin to type, the text simply disappears.

Example:

<input type="text" name="user-name" id="user-name"placeholder="at least 3 characters">

### required

The required attribute doesn’t need much introduction; likeautofocus, it does exactly what you’d expect. By adding it to a form field, the browser requires the user to enter data into that field before submitting the form. This replaces the basic form validation currently implemented with JavaScript, making things a little more usable and saving us a little more development time. required is a Boolean attribute, like autofocus.

Example:

<input type="text" id="given-name" name="given-name"required>