sitepoint

Alexis Goldstein, Estelle Weyl, Louis Lazaris

February 28, 2017

Learning about HTML5 Form Attributes (Part 2)

HTML & CSS

Share this article

sitepoint

The following is an extract from our book, HTML5 & CSS3 for the Real World, 2nd Edition, written by Alexis Goldstein, Louis Lazaris, and Estelle Weyl. Copies are sold in stores worldwide, or you can buy it in ebook form here.

The `pattern` Attribute

sitepoint

form text, you can limit what syntax is acceptable with the `pattern` attribute.

The regular expression language used in patterns is the same Perl-based regular expression syntax as JavaScript, except that the pattern attribute must match the entire value, not just a subset. When including a pattern, you should always indicate to users what is the expected (and required) pattern.

The global `title` attribute has special significance when used in conjunction with the `pattern` attribute. Since browsers currently show the value of the `title` attribute on hover such as a tooltip, include pattern instructions that are more detailed than placeholder text, and which form a coherent statement. That `title` attribute will also be displayed with the browser's default error message in browsers that support native form validation with error messaging, which we'll cover later in this chapter.

Note: Regular Expressions

Regular expressions are a feature of most programming languages that allow developers to specify patterns of characters and check to see if a given string matches the pattern. Regular expressions are famously indecipherable to the uninitiated. For instance, one possible regular expression to check if a string is formatted as a hexidecimal color value is this: `#[A-Fa-f0-9]{6}`.

A full tutorial on the syntax of regular expressions is beyond the scope of this book, but there are plenty of great resources, tutorials, and cheat sheets available online if you'd like to learn. Alternatively, you can search the Web or ask around on forums for a pattern that will serve your purpose.

For a basic example, let's add a `pattern` attribute to the password field in our form. We want to enforce the requirement that the password be at least six characters long with no spaces:

```
<li>
<label for="password">I would like my password to be:</label>
<p>(at least 6 characters, no spaces)</p>
<input type="password" id="password" name="password"
required title="(at least 6 characters, no spaces)" pattern="\S{6,}">
</li>
```

sitepoint

ten characters, for example, would be `\S{6,10}`.

As with the `required` attribute, the `pattern` attribute will prevent the form from being submitted if there is no match for the pattern, and will provide an error message.

If your pattern is not a valid regular expression, it will be ignored for the purposes of validation.

The `pattern` attribute has been supported to some extent in all browsers since Firefox 4, Safari 5, Chrome 10, Opera 11, IE10, and Android 2.3. By "some extent," we mean that while all browsers now support the `pattern` attribute, some—notably Safari and Android through 4.4 allow invalid data to be sent on form submission.

Additionally, similar to the `placeholder` and `required` attributes, you can use the value of this attribute to provide the basis for your JavaScript validation code for nonsupporting browsers.

The `disabled` Attribute

The Boolean `disabled` attribute has been around longer than HTML5 but has been expanded on, to a degree. It can be used with any form control except the new `output` element, and, unlike previous versions of HTML, HTML5 allows you to set the `disabled` attribute on a fieldset and have it apply to all the form elements contained in that fieldset.

Generally, form elements with the `disabled` attribute have the content grayed out by default in the browser. Browsers will prohibit the user from focusing on a form control that has the `disabled`

sitepoint

attribute set. This attribute is often used to disable the submit button until all fields are correctly filled out. You can employ the `:disabled` pseudo-class in your CSS to style disabled form controls, and use either `:enabled` or `:not(:disabled)` pseudo-classes to target form controls that aren't disabled. Form controls with the `disabled` attribute aren't submitted along with the form so their values will be inaccessible to your form processing code on the server side. If you want a form value that users are unable to edit but can still see and submit, use the `readonly` attribute.

The `readonly` Attribute

The `readonly` attribute is similar to the `disabled` attribute: it makes it impossible for the user to edit the form field. Unlike `disabled`,however, the field *can*receive focus and its value is submitted with the form. In a comments form, we may want to include the URL of the current page or the title of the article that is being commented on, letting the user know that we're collecting this data without allowing them to change it:

```
<label for="about">Article Title</label>
<input type="text" name="about" id="about" readonly
↵value="http://www.thehtml5herald.com/register.html">
```

The `multiple` Attribute

The `multiple` attribute, if present, indicates that multiple values can be entered in a form control. While it was available in previous versions of HTML, it only applied to the `select` element. In HTML5, it can be added to `file`, `email`, and `range` input types as well. If present, the user can select more than one file, include several comma-separated email addresses, or have a range with two sliders.

While `multiple` file input is supported in all browsers since mobile Safari 7 and IE10, the `multiple` attribute on range input is yet to be supported anywhere at the time of writing.

Note: Spaces or commas?

sitepoint

emails with spaces along with the required comma. Originally the spaces were disallowed in some browsers, but adding spaces after the comma separator has been included in the specification.

The `form` Attribute

Not to be confused with the `form` element, the `form` *attribute* in HTML5 allows you to associate `form` elements with forms in which they're not nested. It means that you can now associate a fieldset or form control with any other form in the document. This solves the age-old issue of forms not being nestable. While you're still unable to nest forms, you can associate "nested" form controls with a form that's not an ancestor.

The `form` attribute takes as its value the ID of the form element with which the fieldset or control should be associated.

If the attribute is omitted, the control will only be submitted with the form in which it's nested. If you include the `form` attribute and remove it, make sure to use `el.removeAttribute('form')` and not `el.setAttribute('form', '')`. If the `form` attribute is included but the value is either empty or points to an invalid form ID, the form control will be disassociated from all forms on the page and will not be submitted with any form, including any ancestral form in which it may be nested.

his attribute is supported in all browsers, starting with Android 4 and IE 11.

sitepoint

The `autocomplete` attribute specifies whether the form, or a form control, should have autocomplete functionality. For most form fields, this will be a drop-down that appears when the user begins typing. For password fields, it's the ability to save the password in the browser. Support for this attribute has been present in browsers for years, though it was never in the specification until HTML5.

If the `autocomplete` attribute is omitted from the form control or the form, the default value is `on` . You may have noticed this the last time you filled out a form. In order to disable autocomplete on a form control (or form), use `autocomplete="off"` . This is a good idea for sensitive information, such as a credit card number, or data that will never need to be reused, such as a CAPTCHA.

Autocompletion is also controlled by the browser, ignoring developer-set preferences. While the default value is `on` , the browser must have it enabled for it to work at all; however, setting the `autocomplete` attribute to `off` overrides the browser's `on` preference for the relevant form control.

The `datalist` Element and the `list` Attribute

Datalists are currently supported in all browsers except Safari, starting with IE10 and Android 4.4.3. In the default form, they fulfill a common requirement: a text field with a set of predefined autocomplete options. Unlike the `select` element, users can enter whatever value they like, but they'll be presented with a set of suggested options.

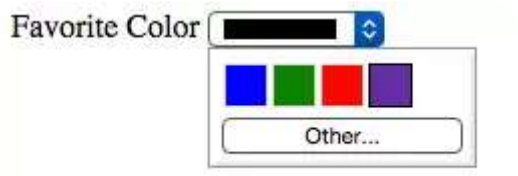

Figure 4.4. The datalist element in action in Firefox

For some input types, like `text` and `date` input types, a drop-down list of options is presented as users type into the field, as shown in Figure 4.4. For the `range` input type, the browser will display little tick marks along the slider rule indicating where suggested options are found. For the `color` input type, users are presented with swatches of color suggestions, with the option to switch to the device's default color picker if they prefer to pick a different color.

sitepoint

attribute on the input. The `list` attribute takes as its value the `id` attribute of the `datalist` you want to associate with the input. One `datalist` can be associated with several input fields.

Here's what this would look like in practice:

```
<label for="favcolor">Favorite Color</label>
<input type="color" list="colors" id="favcolor" name="favcolor">

<datalist id="colors">
<option value="#0000FF" label="blue">
<option value="#008000" label="green">
<option value="#ff0000" label="red">
<option value="#663399" label="RebeccaPurple"> </datalist>
```

The `autofocus` Attribute

The Boolean `autofocus` attribute specifies that a form control should be focused as soon as the page loads. Only one form element can have autofocus in a given page. For a better user experience and for accessibility reasons, it is best *not* to use this attribute.

The input elements support many more attributes, with some being type-specific. The attributes include `alt` , `src` , `height` , and `width` for the image input type, and `step` , `min` , and `max` for numeric input types, including dates and range. `dirname` helps tell the server the form control's directionality. `formaction` , `formenctype` , `formmethod` `formnovalidate` , and `formtarget` provide methods to override the form's attributes `inputmode` helps indicate to the browser what keypad to show when the device is capable of displaying dynamic keypads. `minlength` and `maxlength` dictate the length of allowable input. `checked` , `name` , `size` , `type` , and `value` should already be familiar to you, though `:checked` and `:default` pseudo-classes may be new. We'll cover some of these with their relevant input types next.

Frequently Asked Questions (FAQs) about HTML5 Form Attributes

sitepoint

HTML5 form attributes are properties that provide additional information about an HTML element. They include 'autocomplete', 'autofocus', 'disabled', 'form', 'formaction', 'formenctype', 'formmethod', 'formnovalidate', 'formtarget', 'height and width', 'list', 'min and max', 'multiple', 'pattern', 'placeholder', 'required', 'step', and 'value'. Each attribute serves a unique purpose in form validation, enhancing user experience, and ensuring data integrity.

How does the 'autocomplete' attribute work in HTML5?

The 'autocomplete' attribute in HTML5 is used to control the browser's autocomplete feature for form fields. When set to 'on', the browser will automatically complete the input based on previously entered values. Conversely, when set to 'off', the autocomplete feature is disabled. This attribute is particularly useful for forms that require repetitive information, such as address or email fields.

What is the purpose of the 'autofocus' attribute in HTML5?

The 'autofocus' attribute is used to automatically focus the cursor on a particular input field when a page loads. This can be particularly useful for forms where users are expected to input data immediately upon page load, such as a search bar or login form.

How does the 'disabled' attribute function in HTML5?

The 'disabled' attribute is used to disable an input field, preventing users from interacting with it. This can be useful in situations where certain input fields should not be editable until specific conditions are met.

What is the role of the 'formaction' attribute in HTML5?

The 'formaction' attribute in HTML5 specifies the URL where the form data is sent when the form is submitted. This attribute overrides the 'action' attribute of the 'form' element.

How does the 'pattern' attribute enhance form validation in HTML5?

The 'pattern' attribute in HTML5 is used to specify a regular expression that the input field's value is checked against. This attribute is used for form validation, ensuring that the data entered by the user matches the expected format.

sitepoint

The 'placeholder' attribute in HTML5 is used to provide a hint or example value for the input field. This hint is displayed in the input field before the user enters a value and disappears once the user starts typing.

How does the 'required' attribute function in HTML5?

The 'required' attribute in HTML5 is used to specify that an input field must be filled out before the user can submit the form. If a user attempts to submit a form with an empty required field, a message will be displayed, prompting the user to fill in the field.

What is the role of the 'step' attribute in HTML5?

The 'step' attribute in HTML5 is used with numeric input types to specify the legal number intervals for an input field. For example, if the step attribute is set to '2', the user can only input even numbers.

How does the 'value' attribute work in HTML5?

The 'value' attribute in HTML5 is used to specify the initial value of an input field. This attribute can be used with various input types, including text, checkbox, radio, and button. The value attribute is particularly useful when you want to pre-fill certain form fields for the user.

Alexis Goldstein View Author

Alexis Goldstein first taught herself HTML while a high school student in the mid-1990s, and went on to get her degree in Computer Science from Columbia University. She runs her own software development and training company, aut faciam LLC. Before striking out on her own, Alexis spent seven years in Technology on Wall Street, where she worked in both the cash equity and equity derivative spaces at three major firms, and learned to love daily code reviews. She is a teacher and a co-organizer of Girl Develop It, and a very proud member of the NYC Resistor hackerspace in Brooklyn, NY.

Estelle Weyl View Author

sitepoint

with millions of visitors. Her passion is teaching web development so you'll find her speaking about CSS3, HTML5, JavaScript, and mobile web development at conferences around the United States.

Louis Lazaris View Author

Louis is a front-end developer, writer, and author who has been involved in the web dev industry since 2000. He blogs at Impressive Webs and curates Web Tools Weekly, a newsletter for front-end developers with a focus on tools.

book book excerpt CSS extract html

Share this article

Read Next

Behind the Scenes: A Look at SitePoint's Peer Review Program

Nilson Jacques

A Solid Understanding of Negative Space

James George

5 Ways to Truly Personalize Your "About Me" Page

Gabrielle Gosha

sitepoint

Behind the Scenes with Hover and WebGL

Josh Holmes

6 Great Firefox Extensions for Designers

Ada Ivanoff

Interview with Kris Borchers, JS Foundation Executive Director

Elio Qoshi

How to Build a Better Button in CSS3

Craig Buckler

Why I Love Programming in Go

Mal Curtis

sitepoint

Newsletters

Learning paths

Library

Forums

FAQ

Publish your book with us

Write an article with us

Advertise

About

Our story

Corporate memberships

Terms of use

Privacy policy

Connect

RSS

Facebook

Instagram

Twitter (X)

Back to top