HTML: HTML5 Form Validation Examples

The option of using pure HTML, sometimes with a touch of CSS, to complement JavaScript form validation was until recently unthinkable. Sure there have been all kinds of whacky plug-ins over the years aimed at achieving this, but never a single standard that we could work towards.

For a more detailed introduction to HTML5 form validation you can find some great articles linked under References below. In this article we intend to present only a number of simple examples to get you started, covering the basic form elements.

Before you ask, and someone always does, these examples will currently work in the following browsers: Safari 5, Chrome 6, Opera 9, Firefox 4 Beta and the iPhone/iPad. Also each browser has a slightly different default behaviour.

1. The 'required' attribute

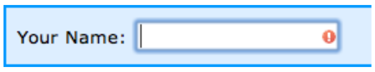
The simplest change you can make to your forms is to mark a text input field as 'required':

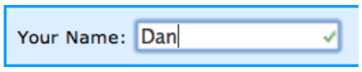
```
Your Name: <input type="text" name="name" required>
```

Your Name:

This informs the (HTML5-aware) web browser that the field is to be considered mandatory. Different browsers may mark the input box in some way (Firefox 4 Beta adds a red box-shadow by default), display a warning (Opera) or even prevent the form from being submitted if this field has no value. Hopefully these behaviours will converge in future releases.

For these examples we have created our own valid/invalid CSS formatting to override the browser default. More on that later. That's why you may see something like the following:


Your Name:


Your Name: Dan

Before you type anything into the box a red marker is shown. As soon as a single character has been entered this changes to a green marker to indicate that the input is 'valid'.

Using CSS you can place markers inside or alongside the input box, or simply use background colours and borders as some browsers do by default.

The `required` attribute can also apply to checkboxes which we've covered separately.

2. New text INPUT types

This is where HTML5 really gets interesting and more useful. Along with the `text` input type, there are now a host of other options, including `email`, `url`, `number`, `tel`, `date` and many others.

On the iPhone/iPad the different input types are associated with different keyboards, making it easier for people to complete your online forms. In other web browsers they can be used in combination with the `required` attribute to limit or give advice on allowable input values.

INPUT type="email"

By changing the input type to `email` while also using the `required` attribute, the browser can be used to validate (in a limited fashion) email addresses:

```
Email Address: <input type="email" name="email" required placeholder="Enter a valid email address">
```

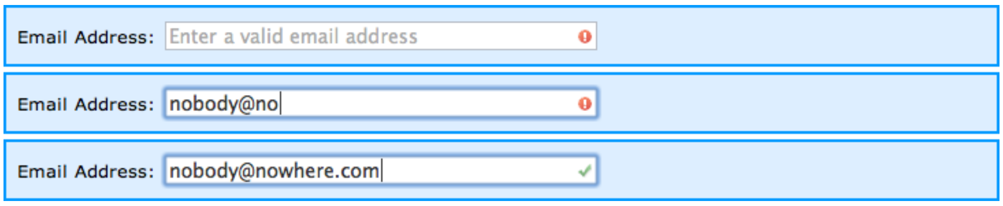
Note that for this example we've made use of another HTML5 attribute `placeholder` which lets us display a prompt or instructions inside the field - something that previously had to be implemented using messy `onfocus` and `onblur` JavaScript events.

The above code displays an input box as follows:

Email Address: Enter a valid email address

Again, different browsers implement this differently. In Opera it's sufficient to enter just `*@*` for the input to be accepted. In Safari, Chrome and Firefox you need to enter at least `*@-.-`. Obviously neither example is very limiting, but it will prevent people from entering completely wrong values, such as phone number, strings with multiple '@'s or spaces.

Here is how it appears in Safari (with our CSS formatting to show the (in)valid state):



Email Address: Enter a valid email address

Email Address: nobody@no

Email Address: nobody@nowhere.com

INPUT type="url"

In a similar fashion to the `email` input type above, this one is designed to accept only properly-formatted URLs. Of course it currently does nothing of the kind, but later you will see how to improve it's behaviour using the `pattern` attribute.

```
Website: <input type="url" name="website" required>
```

Again, the input box appears as normal:

Website:

This time the minimum requirement for most browsers is one or more letters followed by a colon. Again, not very helpful, but it will stop people trying to input their email address or other such nonsense.

As mentioned above, we can improve on this by making use of the `pattern` attribute which accepts a JavaScript regular expression. So the code above becomes:

```
Website: <input type="url" name="website" required pattern="https?://.+">
```

Now our input box will only accept text starting with `http://` or `https://` and at least one additional character:

Website: starting with http

If you're not yet familiar with regular expressions, you really should make it a priority to learn [https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/RegExp] . For those already familiar, note that the ^ and $ are already implicit so that the input has to match the entire expression. Pattern modifiers are not supported [http://www.whatwg.org/specs/web-apps/current-work/multipage/common-input-element-attributes.html#the-pattern-attribute] .

If anyone wants to contribute a more thorough expression to test for valid email or url format, feel free to post it using the Feedback form.

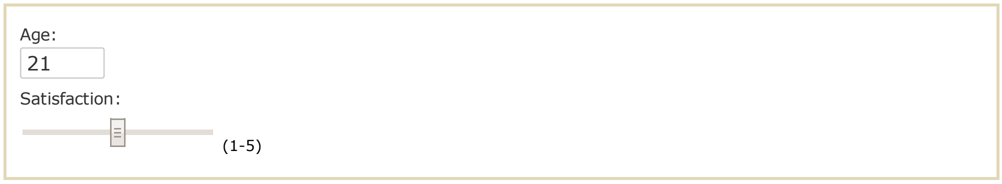
INPUT type="number" and type="range"

The `number` and `range` input types also accept parameters for `min`, `max` and `step`. In most cases you can leave out `step` as it defaults to 1.

Here you see an example including both a `number` input, typically displayed as a 'roller' and a `range` input displayed as a 'slider':

```
Age: <input type="number" size="6" name="age" min="18" max="99" value="21"><br>
Satisfaction: <input type="range" size="2" name="satisfaction" min="1" max="5" value="3">
```

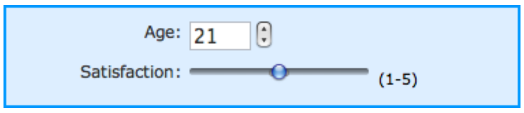
As with other HTML5 input types, browsers that don't recognise the new options will default to simple text inputs. For that reason it's a good idea to include a `size` for the input box.



Age:
21
Satisfaction:
(1-5)

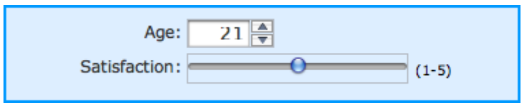
The `slider` option is a bit bizarre in that no values are displayed, but may be useful for more 'analog' inputs. There are some bugs with the `number` input in that if you don't set a `max` value, clicking 'down' with the input blank will result in a very large number.

Here is how the two inputs are displayed in Safari:



Age: 21
Satisfaction: (1-5)

and in Opera:



They are currently not supported in Firefox 4 Beta.

If you want to restrict the input of a text field to numbers without having the up/down arrows associated with the input box, you can always just set the input type to `text` and use a `pattern` of "\d+" (one or more numbers).

INPUT type="password"

We have a separate article with details on validating passwords using HTML5, including JavaScript code for customising the browser generated alert messages.

3. Other HTML5 INPUT types

Other HTML5 input types include:

- color
- date
- datetime
- datetime-local
- month
- search
- tel
- time
- week

The `search` input will, in some browsers, change the styles to match the browser or operating system default search field format. You can see this demonstrated in the Search input above.

The `tel` input type is handy for the iPhone as it selects a different input keyboard. There is no pattern-matching set by default so you would have to implement that yourself using the `pattern` attribute to accept only certain characters.

The `color` input is meant to let you select a hex-code from a colour wheel - or similar - but as yet doesn't appear to have been implemented in the wild.

The other `date`- and `time`-related options do have an effect at least in Opera, with pop-up calendars and other devices appearing to assist with input. While it would be great to see something like this in every browser, for now you probably need to stick with the ubiquitous JavaScript plugins.

4. Styling valid/invalid inputs using CSS

While the code we're using is slightly more complicated, this should get you started:

```
input:required:invalid, input:focus:invalid {
  /* insert your own styles for invalid form input */
  -moz-box-shadow: none;
}
```

The first set of styles can be used to mark an input box as 'invalid', by adding an icon, colouring the text or borders or similar. It will apply to inputs that are `required` but empty, or to inputs that have a required format/pattern which hasn't yet been met.

The `-moz-box-shadow` style is there just to prevent Firefox 4 Beta from adding it's default red border.

For inputs that are both `required` and 'valid' you can use the following:

```
input:required:valid {
  /* insert your own styles for valid form input */
}
```

Some of the articles below, particularly the first two, provide other style/scripting options and solutions for supporting older browsers.

5. Sample styling using images and sprites

As shown above, once you've added HTML5 attributes to your form elements, they can be easily styled using CSS so that each input field is clearly marked as valid or invalid.

```
<style type="text/css">

  input:required:invalid, input:focus:invalid {
    background-image: url(/images/invalid.png);
    background-position: right top;
    background-repeat: no-repeat;
  }
  input:required:valid {
    background-image: url(/images/valid.png);
    background-position: right top;
    background-repeat: no-repeat;
  }

</style>
```

Here you can see the above styles applied to a `required` input field:

Your Name: (required)

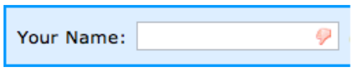
This solution is still more complicated than it needs to be as it requires two extra images to be loaded. Fortunately, we can assume that all browsers supporting HTML5 form validation techniques will also support images being replaced in the CSS by 'Base64 encoded datasets'.

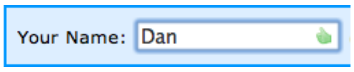
Using a service such as Spritebaker [http://www.stylebaker.com/] or other techniques, the above style settings become:

```
<style type="text/css">

  input:required:invalid, input:focus:invalid {
    background-image: url(data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAABAAAAAQCAYAAAAf8/9hAAAAGXRFWHRTb2Z0
    background-position: right top;
    background-repeat: no-repeat;
    -moz-box-shadow: none;
  }
  input:required:valid {
    background-image: url(data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAABAAAAAQCAYAAAAf8/9hAAAAGXRFWHRTb2Z0
    background-position: right top;
    background-repeat: no-repeat;
  }

</style>
```

The above code can now be copied directly to your CSS style sheet. There's no need to copy any images and, especially if your style-sheets are gzip-compressed, there will be next to no impact on load times. In a few minutes you could have your whole website updated.

For the browser-impaired, this is how the required input field will appear in Safari with either the image or the `Data URI` backgrounds:


Your Name:


Your Name: Dan

The same styling can be extended to `textarea` elements, but won't work for checkboxes, select elements, etc. For those you might want to place the valid/invalid markers alongside the element or format the input elements themselves using borders, background colours, etc.

6. Fallback for the placeholder attribute

The following JavaScript, placed or included at the end of the page, should enable support for the `placeholder` attribute in INPUT fields at least for Internet Explorer 8+, Firefox and Opera:

```
<script type="text/javascript">

  // ref: http://diveintohtml5.org/detect.html [http://diveintohtml5.org/detect.html]
  function supports_input_placeholder()
  {
    var i = document.createElement('input');
    return 'placeholder' in i;
  }

  if(!supports_input_placeholder()) {
    var fields = document.getElementsByTagName('INPUT');
    for(var i=0; i < fields.length; i++) {
      if(fields[i].hasAttribute('placeholder')) {
        fields[i].defaultValue = fields[i].getAttribute('placeholder');
        fields[i].onfocus = function() { if(this.value == this.defaultValue) this.value = ''; }
        fields[i].onblur = function() { if(this.value == '') this.value = this.defaultValue; }
      }
    }
  }

</script>
```

7. INPUT patterns for different data types

URL input pattern:

```
input type="url" pattern="https?://.+"
```

IPv4 Address input pattern:

```
input type="text" pattern="\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}"
```

Date input pattern (dd/mm/yyyy or mm/dd/yyyy):

```
input type="text" pattern="\d{1,2}/\d{1,2}/\d{4}"
```

Price input pattern:

```
input type="text" pattern="\d+(\.\d{2})?"
```

Latitude/Longitude input pattern:

```
input type="text" pattern="-?\d{1,3}\.\d+"
```

Feel free to send any patterns you find useful using the Feedback form.

8. References

- A Form of Madness - Dive Into HTML5 [http://diveintohtml5.org/forms.html]
- A List Apart: Articles: Forward Thinking Form Validation [http://www.alistapart.com/articles/forward-thinking-form-validation/]
- A List Apart: Articles: Inline Validation in Web Forms [http://www.alistapart.com/articles/inline-validation-in-web-forms/]
- Have a Field Day with HTML5 Forms [http://24ways.org/2009/have-a-field-day-with-html5-forms]
- HTML5 input types | 456 Berea Street [http://www.456bereastreet.com/archive/201004/html5_input_types/]
- W3C: Forms - HTML5 [http://www.w3.org/TR/html5/forms.html#client-side-form-validation]

9. Related Articles - Form Validation

- **HTML5 Form Validation Examples** [HTML]
- Validating a checkbox with HTML5 [HTML]
- Password Validation using regular expressions and HTML5 [JAVASCRIPT]
- Date and Time [JAVASCRIPT]
- Preventing Double Form Submission [JAVASCRIPT]
- Form Validation [JAVASCRIPT]
- Credit Card numbers [JAVASCRIPT]
- A simple modal feedback form with no plugins [JAVASCRIPT]
- Protecting forms using a CAPTCHA [PHP]
- Basic Form Handling in PHP [PHP]

10. User Comments

Most recent 20 of 26 comments:

Aaron 6 December, 2014

I REALLY like the way the menu pops out at the top of the page. I've never seen that. Fantastic idea!

Wayne [http://www.behance.net/waynesstewart] 18 September, 2014

Great post. Been looking for some HTML5 validation techniques and this really helped.

Conor 30 April, 2014

Great article, thanks!

Anyone out there know how to adjust the url validation so that it will accept inputs in the following format: www.website.com ? i.e. no need to force a user to input http:// or https://

> You can find a comparison of some interesting regexes for validating URLs here [http://mathiasbynens.be/demo/url-regex]. You just need to pick one and then remove the portion that detects the protocol (xxx://). But as you see, lots of strange looking URLs are actually valid. 😎

Bongani Macheke 8 November, 2013

I used

/^[A-Za-z0-9._%-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}$/

to verify email addresses and it works!

> As others have pointed out, this excludes some valid addresses - usernames including an apostrophe for example. Also some TLDs are longer than four characters.

Matthew Cunliffe [http://www.euro-point.co.uk/] 20 October, 2013

Arjen,

You're correct: and it's not just the plus sign (+), although I have to admit, I've never seen that used in an email address before.

Wikipedia has a list of potentially valid email formats here:
en.wikipedia.org/wiki/Email_address#Valid_email_addresses
[https://en.wikipedia.org/wiki/Email_address#Valid_email_addresses]

Joke, Meepeek 19 August, 2013

I tested the email validation on both Safari and Chrome. It's not working correctly. Typing 'nobody@no' should not pass the validation but it did.

> Most browsers accept xxx@yyy as valid for email input as it can be technically correct in some situations - on an intranet for example. If you want something more restrictive you can add a 'pattern' attribute. e.g
> `pattern="[^ @]+@[^ @]+.[a-z]+"`

Arjen 30 May, 2013

Matthew Cunliffe & Mike, your email regexes are flawed. I have a vaild address with a plus sign (+) in it which will be rejected by your regex (and, as I experience, by many many websites).

Carl 27 March, 2013

Thanks for an interesting post.
Q. How do you only show the fields as 'invalid' after the user leaves (blurs?) the field (or after they've hit send). It's ugly to have ready icons displayed when the page loads. thx

Daniel 5 February, 2013

I can't get the form validation working in Safari 6.0.1. I can't submit the form but I get no error message. It works here with your red/green symbols though. Any suggestions?

> Safari doesn't display any HTML5 validation messages, but it may prevent the form from submitting if any "required" fields are left blank. The messages will appear in Firefox and Opera.
>
> The red/green symbols are applied using CSS and do work in Safari, but are only an indication of whether the input for that field is valid.

email validation [http://www.javaexperience.com/email-and-url-validation-using-html5/] 10 January, 2013

You forgot the most important part: by having these standard types to identify the fields, browsers can provide helpful autofill interfaces. When you select a “tel” or “email” field on your mobile browser, it could open your address book for you to pick a phone number or address from.

James Allardice [http://jamesallardice.com/] 26 September, 2012

The fallback for the placeholder attribute in this article is far from accurate. The main problem is that if the user doesn't enter a new value, the placeholder text will be submitted along with the form. I would highly recommend using one the various placeholder polyfill scripts if you want to support the placeholder attribute in older browsers: e.g. github.com/jamesallardice/Placeholders.js [https://github.com/jamesallardice/Placeholders.js]

Mike 25 September, 2012

Simple telephone (US):

pattern="[0-9]{3}-[0-9]{3}-[0-9]{4}"

Mike 25 September, 2012

My email regex:

<input type="email" name="email" pattern="^\w+([.-]?\w+)*@\w+([.-]?\w+)*(.\w{2,3})+$">

sadaf Majid [http://www.twinexcel.com/] 14 August, 2012

wow very nice n comprehensive overview, i tried few of them like required, date etc but these attributes are not working on IE for this what i can do?

> There are some JavaScript and jQuery 'polyfill' libraries that emulate HTML5 support in Internet Explorer. Try searching for "Modernizr".

Kzqai [http://bitlucid.com/] 2 August, 2012

Wish there were some demos of the tel type and others to test them out.

> The "tel" type seems to only affect the input keyboard for iOS and perhaps similar devices. The "date" input AFAIK has only been implemented in Opera, but hopefully some day there will be cross-browser

support for all the new types. 😎

mivpljiapur 14 July, 2012

you can place markers inside or alongside the input box, or simply use background colours and borders as some browsers do by default.

Matthew Cunliffe [http://www.euro-point.co.uk/] 30 March, 2012

The pattern I use to validate is email addresses is as follows:

([A-Za-z0-9_\-\.])+\@([A-Za-z0-9_\-\.])+\.([A-Za-z]{2,4})

Marco Berrocal [http://www.webmentor.cr/] 9 March, 2012

Keep forgetting to use them. Awersome tutorial and many thanks!!

Jack 18 October, 2011

@Eric Leads

If you're relying strictly on client-side validation for form validation, you're doing it completely wrong. Personally, HTML5 form validation is just as reliable as client-side validation using JavaScript.

JeramieH 12 October, 2011

It's probably obvious but worth being repeated: never trust the client. Yes, this will be great for 99% of the cases, but hackers will intentionally submit data via manual tools just to mess with your system. So always validate on the server as well, just to be certain.

show all 26 comments