Form validation with jQuery

.validate()

validate([options]) *Returns: Validator*

Description: *Validates the selected form.*

validate([options])

options
Type: Object

debug (default: `false`)
Type: Boolean
Enables debug mode. If true, the form is not submitted and certain errors are displayed on the console (will check if a `window.console` property exists). Try to enable when a form is just submitted instead of validation stopping the submit.

Example: Prevents the form from submitting and tries to help setting up the validation with warnings about missing methods and other debug messages.

```
1 $(".selector").validate({
2   debug: true
3 });
```

---

submitHandler (default: `native form submit`)

Type: Function()
Callback for handling the actual submit when the form is valid. Gets the form as the only argument. Replaces the default submit. The right place to submit a form via Ajax after it is validated.

Example: Submits the form via Ajax when valid.

```
$(".selector").validate({
  submitHandler: function(form) {
    $(form).ajaxSubmit();
  }
});
```

Example: Use submitHandler to process something and then using the default submit. Note that "form" refers to a DOM element, this way the validation isn't triggered again.

```
$(".selector").validate({
  submitHandler: function(form) {
    // do other things for a valid form
    form.submit();
  }
});
```

The callback gets passed one argument:

form
Type: Element
The form currently being validated, as a DOMElement.

---

invalidHandler
Type: Function()
Callback for custom code when an invalid form is submitted. Called with an event object as the first argument, and the validator as the second.

Example: Displays a message above the form, indicating how many fields are invalid when the user tries to submit an invalid form.

```
$(".selector").validate({
  invalidHandler: function(event, validator) {
    // 'this' refers to the form
    var errors = validator.numberOfInvalids();
    if (errors) {
      var message = errors == 1
        ? 'You missed 1 field. It has been highlighted'
```

```
        : 'You missed ' + errors + ' fields. They have been highlighte
      $("div.error span").html(message);
      $("div.error").show();
    } else {
      $("div.error").hide();
    }
  }
});
```

The callback gets passed two arguments:

event

Type: Event

A custom event object, since this function is bound as an event handler.

validator

Type: Validator

The validator instance for the current form.

---

ignore (default: `":hidden"`)

Type: Selector

Elements to ignore when validating, simply filtering them out. jQuery's not-method is used, therefore everything that is accepted by not() can be passed as this option. Inputs of type submit and reset are always ignored, so are disabled elements.

Example: Ignores all elements with the class "ignore" when validating.

```
$("#myform").validate({
  ignore: ".ignore"
});
```

---

rules (default: `rules are read from markup (classes, attributes, data)`)

Type: Object

Key/value pairs defining custom rules. Key is the name of an element (or a group of checkboxes/radio buttons), value is an object consisting of rule/parameter pairs or a plain String. Can be combined with class/attribute/data rules. Each rule can be specified as having a depends-property to apply the rule only in certain conditions. See the second example below for details.

Example: Specifies a name element as required and an email element as required (using the shortcut for a single rule) and a valid email address (using another object literal).

```
$(".selector").validate({
  rules: {
    // simple rule, converted to {required:true}
    name: "required",
    // compound rule
    email: {
      required: true,
      email: true
    }
  }
});
```

Example: Specifies a contact element as required and as email address, the latter depending on a checkbox being checked for contact via email.

```
$(".selector").validate({
  rules: {
    contact: {
      required: true,
      email: {
        depends: function(element) {
          return $("#contactform_email").is(":checked");
        }
      }
    }
  }
});
```

Example: Configure a rule that requires a parameter, along with a `depends` callback.

```
$(".selector").validate({
  rules: {
    // at least 15€ when bonus material is included
    pay_what_you_want: {
      required: true
      min: {
        // min needs a parameter passed to it
        param: 15,
        depends: function(element) {
          return $("#bonus-material").is(":checked");
        }
      }
    }
  }
});
```

---

messages (default: `the default message for the method used`)

Type: Object

Key/value pairs defining custom messages. Key is the name of an element, value the message to display for that element. Instead of a plain message, another map with specific messages for each rule can be used. Overrides the title attribute of an element or

the default message for the method (in that order). Each message can be a String or a Callback. The callback is called in the scope of the validator, with the rule's parameters as the first argument and the element as the second, and must return a String to display as the message.

Example: Specifies a name element as required and an email element as required and a valid email address. A single message is specified for the name element, and two messages for email.

```
$(".selector").validate({
  rules: {
    name: "required",
    email: {
      required: true,
      email: true
    }
  },
  messages: {
    name: "Please specify your name",
    email: {
      required: "We need your email address to contact you",
      email: "Your email address must be in the format of name@domain.
    }
  }
});
```

Example: Validates the name-field as required and having at least two characters. Provides a callback message using jQuery.validator.format to avoid having to specify the parameter in two places.

```
$(".selector").validate({
  rules: {
    name: {
      required: true,
      minlength: 2
    }
  },
  messages: {
    name: {
      required: "We need your email address to contact you",
      minlength: jQuery.validator.format("At least {0} characters requ
    }
  }
});
```

groups

Type: Object

Specify grouping of error messages. A group consists of an arbitrary group name as the key and a space separated list of element names as the value. Use errorPlacement to control where the group message is placed.

Example: Use a table layout for the form, placing error messags in the next cell after the input.

```
$("#myform").validate({
  groups: {
    username: "fname lname"
  },
  errorPlacement: function(error, element) {
    if (element.attr("name") == "fname" || element.attr("name") == "lr
      error.insertAfter("#lastname");
    } else {
      error.insertAfter(element);
    }
  }
});
```

onsubmit (default: `true`)

Type: Boolean

Validate the form on submit. Set to false to use only other events for validation.

Set to a Function to decide for yourself when to run validation.

A boolean true is not a valid value.

Example: Disables onsubmit validation, allowing the user to submit whatever he wants, while still validating on keyup/blur/click events (if not specified otherwise).

```
$(".selector").validate({
  onsubmit: false
});
```

onfocusout

Type: Boolean or Function()

Validate elements (except checkboxes/radio buttons) on blur. If nothing is entered, all rules are skipped, except when the field was already marked as invalid.

Set to a Function to decide for yourself when to run validation.

A boolean true is not a valid value.

Example: Disables onblur validation.

```
$(".selector").validate({
  onfocusout: false
});
```

The callback gets passed two arguments:

element

Type: Element

The element currently being validated, as a DOMElement.

---

event

Type: Event

The event object for this focusout event.

---

onkeyup

Type: Boolean or Function()

Validate elements on keyup. As long as the field is not marked as invalid, nothing happens. Otherwise, all rules are checked on each key up event. Set to false to disable.

Set to a Function to decide for yourself when to run validation.

A boolean true is not a valid value.

Example: Disables onkeyup validation.

```
$(".selector").validate({
  onkeyup: false
});
```

The callback gets passed two arguments:

element

Type: Element

The element currently being validated, as a DOMElement.

event

Type: Event

The event object for this keyup event.

---

onclick

Type: Boolean or Function()

Validate checkboxes and radio buttons on click. Set to false to disable.

Set to a Function to decide for yourself when to run validation.

A boolean true is not a valid value.

Example: Disables onclick validation of checkboxes and radio buttons.

```
$(".selector").validate({
  onclick: false
});
```

The callback gets passed two arguments:

element

Type: Element

The element currently being validated, as a DOMElement.

---

event

Type: Event

The event object for this click event.

---

focusInvalid (default: `true`)

Type: Boolean

Focus the last active or first invalid element on submit via validator.focusInvalid(). The last active element is the one that had focus when the form was submitted, avoiding stealing its focus. If there was no element focused, the first one in the form gets it, unless this option is turned off.

Example: Disables focusing of invalid elements.

```
$(".selector").validate({
  focusInvalid: false
```

```
});
```

focusCleanup (default: `false`)

Type: Boolean

If enabled, removes the errorClass from the invalid elements and hides all error messages whenever the element is focused. Avoid combination with focusInvalid.

Example: Enables cleanup when focusing elements, removing the error class and hiding error messages when an element is focused.

```
$(".selector").validate({
  focusCleanup: true
});
```

errorClass (default: `"error"`)

Type: String

Use this class to create error labels, to look for existing error labels and to add it to invalid elements.

Example: Sets the error class to "invalid".

```
$(".selector").validate({
  errorClass: "invalid"
});
```

validClass (default: `"valid"`)

Type: String

This class is added to an element after it was validated and considered valid.

Example: Sets the valid class to "success".

```
$(".selector").validate({
  validClass: "success"
});
```

errorElement (default: `"label"`)

Type: String

Use this element type to create error messages and to look for existing error messages. The default, "label", has the advantage of creating a meaningful link between error message and invalid field using the for attribute (which is always used, regardless of element type).

Example: Sets the error element to "em".

```
$(".selector").validate({
  errorElement: "em"
});
```

---

wrapper (default: `window`)

Type: String

Wrap error labels with the specified element. Useful in combination with errorLabelContainer to create a list of error messages.

Example: Wrap each error element with a list item, useful when using an ordered or unordered list as the error container.

```
$(".selector").validate({
  wrapper: "li"
});
```

---

errorLabelContainer

Type: Selector

Hide and show this container when validating.

Example: All error labels are displayed inside an unordered list with the ID "messageBox", as specified by the selector passed as errorContainer option. All error elements are wrapped inside a li element, to create a list of messages.

```
$("#myform").validate({
  errorLabelContainer: "#messageBox",
  wrapper: "li",
  submitHandler: function() { alert("Submitted!") }
});
```

---

errorContainer

Type: Selector

Hide and show this container when validating.

Example: Uses an additonal container for error messages. The elements given as the errorContainer are all shown and hidden when errors occur. However, the error labels themselves are added to the element(s) given as errorLabelContainer, here an unordered list. Therefore the error labels are also wrapped into li elements (wrapper option).

```
$("#myform").validate({
  errorContainer: "#messageBox1, #messageBox2",
  errorLabelContainer: "#messageBox1 ul",
  wrapper: "li", debug:true,
  submitHandler: function() { alert("Submitted!") }
});
```

---

showErrors

Type: Function()

A custom message display handler. Gets the map of errors as the first argument and an array of errors as the second, called in the context of the validator object. The arguments contain only those elements currently validated, which can be a single element when doing validation onblur/keyup. You can trigger (in addition to your own messages) the default behaviour by calling this.defaultShowErrors().

Example: Update the number of invalid elements each time an error is displayed. Delegates to the default implementation for the actual error display.

```
$(".selector").validate({
  showErrors: function(errorMap, errorList) {
    $("#summary").html("Your form contains "
      + this.numberOfInvalids()
      + " errors, see details below.");
    this.defaultShowErrors();
  }
});
```

The callback gets passed two arguments:

errorMap

Type: Object

Key/value pairs, where the key refers to the name of an input field, values the message to be displayed for that input.

---

errorList

Type: Array
An array for all currently validated elements. Contains objects with the following two properties:

message
Type: String
The message to be displayed for an input.

element
Type: Element
The DOMElement for this entry.

errorPlacement (default: `Places the error label after the invalid element`)
Type: Function()
Customize placement of created error labels. First argument: The created error label as a jQuery object. Second argument: The invalid element as a jQuery object.

Example: Use a table layout for the form, placing error messages in the next cell after the input.

```
$("#myform").validate({
  errorPlacement: function(error, element) {
    error.appendTo( element.parent("td").next("td") );
  }
});
```

The callback gets passed two arguments:

error
Type: jQuery
The error label to insert into the DOM.

element
Type: jQuery
The validated input, for relative positioning.

success
Type: String or Function()
If specified, the error label is displayed to show a valid element. If a String is given, it is added as a class to the label. If a Function is given, it is called with the label (as a jQuery object) and the validated input (as a DOM element). The label can be used to add a text

like "ok!".

Example: Add a class "valid" to valid elements, styled via CSS.

```
$("#myform").validate({
  success: "valid",
  submitHandler: function() { alert("Submitted!") }
});
```

Example: Add a class "valid" to valid elements, styled via CSS, and add the text "Ok!".

```
$("#myform").validate({
  success: function(label) {
    label.addClass("valid").text("Ok!")
  },
  submitHandler: function() { alert("Submitted!") }
});
```

The callback gets passed two arguments:

label

Type: jQuery

The error label. Use to add a class or replace the text content.

---

element

Type: Element

The element currently being validated, as a DOMElement.

---

highlight (default: `Adds errorClass (see the option) to the element`)

Type: Function()

How to highlight invalid fields. Override to decide which fields and how to highlight.

Example: Highlights an invalid element by fading it out and in again.

```
$(".selector").validate({
  highlight: function(element, errorClass) {
    $(element).fadeOut(function() {
      $(element).fadeIn();
    });
  }
});
```

Example: Adds the error class to both the invalid element and its label

```
$(".selector").validate({
  highlight: function(element, errorClass, validClass) {
    $(element).addClass(errorClass).removeClass(validClass);
    $(element.form).find("label[for=" + element.id + "]")
      .addClass(errorClass);
  },
  unhighlight: function(element, errorClass, validClass) {
    $(element).removeClass(errorClass).addClass(validClass);
    $(element.form).find("label[for=" + element.id + "]")
      .removeClass(errorClass);
  }
});
```

The callback gets passed three arguments:

element

Type: Element

The invalid DOM element, usually an `input`.

errorClass

Type: String

Current value of the `errorClass` option.

validClass

Type: String

Current value of the `validClass` option.

unhighlight (default: `Removes the errorClass`)

Type: Function()

Called to revert changes made by option highlight, same arguments as highlight.

ignoreTitle (default: `false`)

Type: Boolean

Set to skip reading messages from the title attribute, helps to avoid issues with Google Toolbar; default is false for compability, the message-from-title is likely to be completely removed in a future release.

Example: Configure the plugin to ignore title attributes on validated elements when looking for messages.

```
$(".selector").validate({
  ignoreTitle: true
});
```

This method sets up event handlers for submit, focus, keyup, blur and click to trigger validation of the entire form or individual elements. Each one can be disabled, see the onxxx options (onsubmit, onfocusout, onkeyup, onclick). focusInvalid focuses elements when submitting an invalid form.

Use the debug option to ease setting up validation rules, it always prevents the default submit, even when script errors occur.

Use submitHandler to implement your own form submit, eg. via Ajax. Use invalidHandler to react when an invalid form is submitted.

Use rules and messages to specify which elements to validate, and how. See rules() for more details about specifying validation rules.

Use errorClass, errorElement, wrapper, errorLabelContainer, errorContainer, showErrors, success, errorPlacement, highlight, unhighlight, and ignoreTitle to control how invalid elements and error messages are displayed.

This entry was posted in Plugin Methods on May 23, 2013 [http://jqueryvalidation.org/validate/] by jzaefferer.