Really easy field validation

Here's a form validation script that is very easy to use.

Current Version: 1.5.4.1 - 06 Jan 2007 - Demo / Download

Instructions

The basic method is to attach to the form's `onsubmit` event, read out all the form elements' classes and perform validation if required. If a field fails validation, reveal field validation advice and prevent the form from submitting.

Include the javascript libraries:

```
<script src="prototype.js" type="text/javascript"></script>
<script src="validation.js" type="text/javascript"></script>
```

You write elements like this:

```
<input class="required validate-number" id="field1" name="field1" />
```

passing the validation requirements in the `class` attribute.

You then activate validation by passing the form or form's `id` attribute like this:

```
<script type="text/javascript">
     new Validation('form-id'); // OR new Validation(document.forms[0]);
</script>
```

It has a number of tests built-in but is extensible to include your custom validation checks.

The validator also avoids validating fields that are hidden or children of elements hidden by the CSS property `display:none`. This way you can give a field the class of 'required' but it's only validated if it is visible on the form. The demo illustrates what I am talking about

Options

Here's the list of classes available to add to your field elements:

- `required` (not blank)
- `validate-number` (a valid number)

- `validate-digits` (digits only)
- `validate-alpha` (letters only)
- `validate-alphanum` (only letters and numbers)
- `validate-date` (a valid date value)
- `validate-email` (a valid email address)
- `validate-url` (a valid URL)
- `validate-date-au` (a date formatted as; dd/mm/yyyy)
- `validate-currency-dollar` (a valid dollar value)
- `validate-selection` (first option e.g. 'Select one...' is not selected option)
- `validate-one-required` (At least one textbox/radio element must be selected in a group - see below*)

*To use the `validate-one-required` validator you must first add the class name to only one checkbox/radio button in the group (last one is probably best) and then place all the input elements within a parent element, for example a div element. That way the library can find all the checkboxes/radio buttons to check and place the validation advice element at the bottom of the parent element to make it appear after the group of checkboxes/radio buttons.

When the validation object is initialised you can pass the option `{stopOnFirst : true}` to enable the stop on first validation failure behaiour. The demo above has this set to false which is the default. If set to true only the first validation failure advice will be displayed when the form is submitted instead of all at once.

```
<script type="text/javascript">
     new Validation('form-id',{stopOnFirst:true});
</script>
```

You can also pass the option `{immediate : true}` to enable field valiation when leaving each field. That is on the onblur event for all the form elements.

By default the library will add an event listener to the form's onsubmit event and stop the event if the validation fails. If you pass the option `{onSubmit : false}` it wont do that. This way you can call the validate function manually within your own javascript.

By default the library will focus on the first field that contains an error. If you pass the option `{focusOnError : false}` it wont do that.

You can also pass the option `{useTitles : true}` to make the field validators use the form elements' title attribute value as the error advice message.

You can set callbacks by using the options
`{onFormValidate : yourFunction, onElementValidate : yourFunction}`.

onFormValidate is called after form validation takes place and takes two arguments: the validation result (true or false) and a reference to the form. OnElementValidate is called after each form element is validated and also takes 2 arguments: the validation result (true or false) and a reference to the form element.

Instead of using the error message in the validator you can create your own validation advice page element. Now when the script is creating the advice element it first looks for an element with an id matching `'advice-' + validation-class-name + '-' + element.id` and if not found then one matching `'advice-' + element.id`. If your form element does not have an id attribute then match the name attribute. If it finds an element it will make that one appear. See the 'Donation' field in the demo for an example. If you make a custom validation advice element make sure you set the style to `display : none` .

Also if you reference the effects.js file from Scriptaculous in your page head, it'll use a fade-in effect for the validation advice.

```
<script src="effects.js" type="text/javascript"></script>
```

CSS Hooks

As well as the validation css classes above, the validation library uses CSS classes to indicate validation status:

`validation-failed` and `validation-passed`

The validation advice element has a class of `validation-advice` and an id matching the following pattern

`'advice-' + validation-class-name + '-' + element.id`

so if the field '`birthdate`' uses the '`validate-date`' validation class then the validation advice element will have an id of '`advice-validate-date-birthdate`'.

Javascript API

By default the class attaches an event observer to the form's onsubmit event. If you prefer to do the form submit via javascript yourself you can still validate the form like this:

```
<script type="text/javascript">
    var valid = new Validation('form-id', {onSubmit:false});
    var result = valid.validate();
</script>
```

The instance method, validate(), will return true or false.

The class has an instance function which resets all the field validation:

```
<script type="text/javascript">
    var valid = new Validation('form-id');
    valid.reset();
</script>
```

Note that it doesn't reset the form, just the validation.

The Validation class also has some static methods that can be used independantly.

```
Validation.validate([element OR element id] [, options])
```

This validates the field (or field with that id), using all validation classes present. You can also pass the option `{useTitle : true}` to make the field validator use the form element's title attribute value as the error advice message.

You can run a specific validation test on a field or field value by doing this:

```
Validation.get('validator-name').test(value [, element]);
```

To add your own validator do this:

```
Validation.add('class-name', 'Error message text', function(value [, element]
     return /* do validation here */
}, options);
```

or this:

```
Validation.add('class-name', 'Error message text', options);
```

The first example above includes a function as the third argument. The function enables you to write your own custom validation. The options argument is optional. The second example the third argument has become the options argument. Validator options can be used to perform common validation options without the need to write them into a function. Multiple options can be combined to create a complex validator and they can also enhance your custom validation function. Here are the available options and example usage below:

```
Validation.add('class-name', 'Error message text', {
     pattern : new RegExp("^[a-zA-Z]+$","gi"), // only letter allowed
     minLength : 6, // value must be at least 6 characters
     maxLength : 13, // value must be no longer than 13 characters
     min : 5, // value is not less than this number
     max : 100, // value is not more than this number
     notOneOf : ['password', 'PASSWORD'], // value does not equal anything in
     oneOf : ['fish','chicken','beef'], // value must equal one of the values
     is :  '5', // value is equal to this string
     isNot : 'turnip', //value is not equal to this string
     equalToField : 'password', // value is equal to the form element with th
     notEqualToField : 'username', // value is not equal to the form element
     include : ['validate-alphanum'] // also tests each validator included in
});
```

For example here's one of the in-built ones:

```
Validation.add('validate-alpha', 'Please use letters only (a-z) in this field
     return Validation.get('IsEmpty').test(v) ||  /^[a-zA-Z]+$/.test(v)
});
```

And here's a custom one using options:

```
Validation.addAllThese('validate-password', 'Your password must be more than
    minLength : 7,
    notOneOf : ['password','PASSWORD','1234567','0123456'],
    notEqualToField : 'username'
});
```

If you supply a custom function and a combination of options they are all tested and if all are true the field validates.

When you add a new validator it is added to a static group of validation methods with the class name as key. You then must use the class in the form elements to use your custom validation function.

To make adding mupltiple custom validators easier you can use `Validation.addAllThese()` like this:

```
Validation.addAllThese([
     ['required', 'This is a required field.', function(v) {
       return !Validation.get('IsEmpty').test(v);
     }],
     ['validate-number', 'Please use numbers only in this field.', function(v
       return Validation.get('IsEmpty').test(v) || !isNaN(v);
     }],
     ['validate-digits', 'Please use numbers only in this field.', function(v
       return Validation.get('IsEmpty').test(v) ||  !/[^d]/.test(v);
     }]
]);
```

You pass an array, where each element of the array is an array with 3 or 4 elements: [className, error, function, options] or [className, error, options]

Support

Please submit your questions, suggestions, patches, bugs, extra validators and such to the Dexgogo group on Google Groups .

Demo & Download

You can view the demo to see some examples or the demo can be downloaded as a zip file under the MIT License, same as Prototype and Scriptaculous.

Change Log

Version 1.5.4.1

- Oops, left a trailing comma on an array... thanks Ed !

Version 1.5.4

- Added 'validate-selection' for HTML select elements - validation fails of first option is selected value
- Added Validator options for creating validators - good for custom validation combinations if you don't want to write a javascript function - see documentation above.

Version 1.5.3.1

- Fixed problem getting the advice elements caused by change in the behaviour of the Protype $() function - specifically the return value when an element is not found.

Version 1.5.3

- Added small change to better support radio/checkbox elements. The validation advice message is shown at the bottom of the parent element of all the radio/checkboxes; i.e. at the end of the group of elements.
- Added example 'validate-one-required' validator for checkbox/radio elements

Version 1.5.2

- Added 2 callback options: onFormValidate and onElementValidate, thanks for the idea John Farrar
- Fixed URL validator,thanks Bruno
- Fixed advice text on number validators, thanks Ade
- Fixed number validator validating space as valid, thanks Rob McDonagh
- Changed isEmpty validator so that a value of only whitespace is no longer considered empty

Version1.5.1

- Oops

Version 1.5

- Added support for using the title attribute as the error advice text as per xav's idea
- Added support for forms with elements with no id attributes
- The field element is now passed as a reference to the validate function to enable more complex validators
- Added support for multiple form instances for the focusOnError option
- Added URL validator from Marcus Bointon
- Improved date-au validator from Tanvir
- Started using Insertion.After from prototype for compatibility reasons as per xav's suggestion
- Completed some code reorganisation to support future directions

Version 1.4

- Custom field validation advice as per Sidney's idea
- Fixed internal check property name - make consistant with general object property notation i.e. camelcase it
- **Behaviour change**: Error advice nodes are now hidden instead of removed when not needed

Version 1.3

- Added reset function as per Paul Shannon's idea
- Added focusOnError option, thanks Ted Wise for the idea and some demo code
- Fixed a typo, thanks Analgesia!

Version 1.2.1

- Removed $ shortcuts from main code body (There's a worry on the mailing list about polluting the global namespace with an alphabet of $ functions. So, they are commented out in code, you can uncomment them if you want them)

Version 1.2

- New 'immediate:true' option to add onblur validation to form elements. Thanks Mike Rumble !

Version 1.1

- Added new function Validator.addAllThese()
- Validator now respects hidden form elements
- code tidy-up
- Updated demo

Linkography

- Scriptaculous Scriptaculous home page
- Dexgogo group on Google Groups Join the Dexagogo group and spin yarns about your l337 javascript skillz
- MIT License
- thanks Ed on the Dexagogo Google group
- Bruno
- Ade
- Rob McDonagh
- Marcus Bointon Marcus Bointon
- Sidney's
- Paul Shannon's Pauls home page
- Mike Rumble Mike's home page

Tags

- javascript
- prototype

NICE PEOPLE

Adrian Lynch Rosemary Lynch Nathan Tetlaw Kim Davies Dan Woods Port80

AND OF COURSE

My del.icio.us flickr Technorati 43 Things

PROFESSIONAL STUFF

australian web industry association individual member
