Search...

Getting started
Components

M4Q
About
Population
Constructor
Ajax
Animation
Loops
Visibility
Effects
Subtree functions
Attributes
Html, text and value
Css and classes
Position and size
Manipulation
DataSet
Events
Utils

Base
Containers
Grid system
Typography
Tables
Forms
Buttons
Images
Figures
Lists

Form components
Checkbox
File input
Input
Input material
Keypad
Rating
Radio
Select
Slider
Double Slider
Spinner
Switch
Tag input
Textarea

Menus
App bar
Bottom navigation
Bottom sheet
Menu
Ribbon Menu
Side bar
Side navigation

Controls
Accordion
Badge
Carousel
Cards
Cube
Counter
Charms
Chat
Donut
Image compare
Image magnifier
Gravatar
List
ListView
Master
NavView
Panels
Progress & Activity
Streamer
Stepper
Splitter
Tabs
Tabs material
Table
Tiles
TreeView
Wizard

Information
Dialogs
Info box
Hints
Notify system
Popovers
Toasts
Windows

Date & time
Calendar
Calendar picker
Date picker
Time picker
Countdown
Formatting date

Media

Validator

Makes simple client-side form validation easy, whilst still offering plenty of customization options. The widget comes bundled with a useful set of validation methods, including URL, email, hex color, min and max values, length validation.

How to

To activate Validator add attribute `data-role="validator"` to form and define validation functions over attribute `data-validate="..."` for inputs.

Supported elements: `input` , `select` , `textarea` , `checkbox` , `switch` and `radio` . For `checkbox` , `switch` and `radio` you can use only `required` validation function. More about form elements read in this article.

First name
Enter first name
Email
Enter email
Select option
I accept the terms
Type 1 Type 2 Type 3
Submit

```
<form data-role="validator" action="javascript:">
    <div class="row mb-2">
        <div class="cell-md-6">
            <label>First name</label>
            <input type="text"
                data-validate="required minlength=6"
                placeholder="Enter first name">
            <span class="invalid_feedback">
                Input correct name with min length 6 symbols
            </span>
        </div>
        <div class="cell-md-6">
            <label>Email</label>
            <input type="email"
                data-validate="required email"
                placeholder="Enter email" data-role="input">
            <span class="invalid_feedback">
                Input correct email address
            </span>
        </div>
    </div>

    <div class="mt-2 mb-2">
        <label>Select option</label>
        <select data-role="select" data-validate="required not=-1">
            <option value="-1" class="d-none"></option>
            <option value="1">Value 1</option>
            <option value="2">Value 2</option>
            <option value="3">Value 3</option>
        </select>
        <span class="invalid_feedback">
            You must select a option!
        </span>
    </div>

    <div class="row mb-2">
        <div class="cell-md-6">
            <input type="checkbox"
                data-role="checkbox"
                data-caption="I accept the terms"
                data-validate="required">
            <span class="invalid_feedback">
                You must accept this!
            </span>
        </div>
        <div class="cell-md-6">
            <input type="radio" name="__r1"
                data-role="radio" value="1"
                data-validate="required" data-caption="Type 1">
            <input type="radio" name="__r1"
                data-role="radio" value="2"
                data-validate="required" data-caption="Type 2">
            <input type="radio" name="__r1"
                data-role="radio" value="3"
                data-validate="required" data-caption="Type 3">
            <span class="invalid_feedback"
                >You must select a option!
            </span>
        </div>
    </div>

    <button class="button primary">Submit</button>
</form>
```

Validating radio

To validate `radio` component you must add attribute `name` to element and use validate function `required` . Validate function must be added to all radio element.

```
<input type="radio" name="__r1"
    data-role="radio" value="1"
    data-validate="required" data-caption="Type 1">
<input type="radio" name="__r1"
```

Table of contents

```
    data-role="radio" value="2"
    data-validate="required" data-caption="Type 2">
<input type="radio" name="__r1"
    data-role="radio" value="3"
    data-validate="required" data-caption="Type 3">
```

Validating select

To validate `select` component you must add specific first option and use validate functions: `required`, `length`, `minlength`, `maxlength` and `not`.

`length`, `minlength`, `maxlength` - for use with select with `multiple` option.

```
<select data-role="select" data-validate="required not=-1">
    <option value="-1" class="d-none"></option>
    <option value="1">Value 1</option>
    <option value="2">Value 2</option>
    <option value="3">Value 3</option>
</select>
```

Invalid feedback

You can create user-friendly **invalid feedback**. To create it, add element with class `.invalid_feedback` after input.

Email

Enter email

Send

```
<label>Email</label>
<input type="email"
    data-validate="required email"
    placeholder="Enter email" data-role="input">
<span class="invalid_feedback">
    Input correct email address
</span>
```

Functions

Metro 4 Validator supports 12 validation functions. To define validation for input add to input attribute `data-validate="..."`. You can combine validation functions. To set function argument use record `func_name=arg`.

Input name (required)

Input age (required number)

Input pin (required digits length=4)

Input password (required)

Input password again (required compare=pass1)

Submit

```
<form data-role="validator" action="javascript:">
    <div class="form-group">
        <label>Input name</label>
        <input type="text" data-validate="required" name="nickname">
    </div>
    <div class="form-group">
        <label>Input age</label>
        <input type="text" data-validate="required number" name="age">
    </div>
    <div class="form-group">
        <label>Input pin</label>
        <input type="text" data-validate="required digits length=4" name="pin">
    </div>
    <div class="form-group">
        <label>Input password</label>
        <input type="password" data-validate="required" name="pass1">
    </div>
    <div class="form-group">
        <label>Input password again</label>
        <input type="password" data-validate="required compare=pass1" name="pass2">
        <span class="invalid_feedback">
            Pass1 and Pass2 must be equal.
        </span>
    </div>
    <br />
    <button class="button success">Submit</button>
</form>
```

Validate functions

You can use next functions for validating input value: `required`, `length`, `minlength`, `maxlength`, `min`, `max`, `email`, `domain`, `url`, `date`, `number`, `digits`, `hexcolor`, `color`, `pattern`, `compare`, `not`, `notequals` and `custom`.

required - mark field as required. Value can not be empty.

```
<input type="text" data-validate="required">
```

length - set length for value in symbols

```
<input type="text" data-validate="length=4">
```

minlength - set min length for value in symbols

```
<input type="text" data-validate="minlength=4">
```

maxlength: set max length for value in symbols

```
<input type="text" data-validate="maxlength=4">
```

min - value must be great or equal then `min`

```
<input type="text" data-validate="min=4">
```

max: value must be less or equal then `max`

```
<input type="text" data-validate="max=4">
```

email - value must be a valid email

```
<input type="text" data-validate="email">
```

domain - value must be a valid domain name

```
<input type="text" data-validate="domain">
```

url - valid url required

```
<input type="text" data-validate="url">
```

date - valid date required

```
<input type="text" data-validate="date" data-value-format="_format_input_date_">
```

number - value must be a number

```
<input type="text" data-validate="number">
```

integer - value must be a integer number

```
<input type="text" data-validate="integer">
```

float - value must be a float number

```
<input type="text" data-validate="float">
```

digits: value must contains only digits

```
<input type="text" data-validate="digits">
```

hexcolor - value must be a valid hex color

```
<input type="text" data-validate="hexcolor">
```

color - value must contains standard color name

```
<input type="text" data-validate="color">
```

pattern - value must be equal with regex pattern. Currently regex modifiers not supported. Parsing regex modifiers will be added in next version.

```
<input type="text" data-validate="pattern=(^#[0-9A-F]{6}$)|(^#[0-9A-F]{3}$)">
```

compare - value must be equal to compared input

```
<input type="password" data-validate="required" name="pass1">
<input type="password" data-validate="required compare=pass1" name="pass2">
```

not - value must not be equal to compared value

```
<select data-role="select" data-validate="required not=-1">
    <option value="-1" class="d-none"></option>
    <option value="1">Value 1</option>
    <option value="2">Value 2</option>
    <option value="3">Value 3</option>
</select>
```

notequals - value must not be equal to compared input

```
<input type="text" data-validate="required" name="ip_1">
<input type="text" data-validate="required notequals=ip_1" name="ip_2">
```

equals - value must be equal to compared input

```
<input type="text" data-validate="required" name="ip_1">
<input type="text" data-validate="required equals=ip_1" name="ip_2">
```

custom - required valid function exist name

```
--- html
<input type="text" data-validate="custom=myValidateFuncName">

---- javascript
function myValidateFuncName(val){
    val = parseInt(val);
    return Metro.utils.isInt(val) && (val > 3 && val < 9);
}
```

Combine validating functions

You can combine function. To use it, add function to attribute `data-validate` with `space` delimiter.

```
<input type="text" data-validate="required pattern=^[\w]{7,15}$">
```

Functions mode

New in 4.2.6

By default, all validating functions work in **required mode** - field value cannot be empty and must have required value. You can change this behavior with special form attribute `data-required-mode`. If you set this attribute to `false`, functions will work only if field value not empty or with `required` function in the pair.

Default behavior, check value is valid email and not empty

Submit Reset

Required mode - false, check value is valid email when not empty

Submit Reset

Required mode - false, check value is valid email and not empty with rule required

Submit Reset

```
<p class="h5 text-light">
    Default behavior, check value is valid email and not empty
</p>
<form data-role="validator" action="javascript:" class="mt-2">
    <input type="text" data-validate="email">
    <div class="form-actions">
        <button class="button" type="submit">Submit</button>
        <button class="button" type="reset">Reset</button>
    </div>
</form>

<p class="h5 text-light">
    Required mode  - false, check value is valid email when not empty
</p>
<form data-role="validator" data-required-mode="false" action="javascript:" class="mt-2">
    <input type="text" data-validate="email">
    <div class="form-actions">
        <button class="button" type="submit">Submit</button>
        <button class="button" type="reset">Reset</button>
    </div>
</form>

<p class="h5 text-light">
    Required mode  - false, check value is valid email and not empty with rule required
</p>
<form data-role="validator" data-required-mode="false" action="javascript:" class="mt-2">
    <input type="text" data-validate="required email">
    <div class="form-actions">
        <button class="button" type="submit">Submit</button>
        <button class="button" type="reset">Reset</button>
    </div>
</form>
```

Events

When Validator is works, fired any events: `onBeforeSubmit(form, data)` , `onError(element, value)` , `onValidate(element, value)` , `onErrorForm(form, data)` , `onValidateForm(form, data)` , `onSubmit(form, data)` , `onValidatorCreate(form)` .

To use event add attribute `data-on-*` to form.

Your nickname

Enter your nickname

Submit

```
<form data-role="validator" action="javascript:"
      data-on-submit="alert('Your submitted name is: ' + this['nick_name'].value)">
    <div class="row mb-2">
        <div class="cell-md-6">
            <label>Your nickname</label>
            <input type="text" data-validate="minlength=6" name="nick_name">
            <span class="invalid_feedback">
                Input correct name with min length 6 symbols
            </span>
        </div>
    </div>
    <button class="button primary">Submit</button>
</form>
```

Event	Data	Context	Desc
`onError(el, val)`	`data-on-error`	`element`	Event generate if field not passed validation
`onValidate(el, val)`	`data-on-validate`	`element`	Event generate if field passed validation
`onErrorForm(log, el)`	`data-on-error-form`	`form`	Event generate if form not passed validation
`onValidateForm(el)`	`data-on-validate-form`	`form`	Event generate if form passed validation
`onBeforeSubmit(el)`	`data-on-before-submit`	`form`	If this function return false, form will not be submitted
`onSubmit(el)`	`data-on-submit`	`form`	Event generate when form is submitted

Submit form

The following conditions must be met to send the form:

The all the validations must be passed
If an `onBeforeSubmit` event is defined, it must return true

Error handling

If not passed all validations, a form will not be submitted. You can handling the validation errors with validator events: `onError` and `onErrorForm` .

onError

Event `onError` generated for each inputs when input not passed validation.

Input name (required)

Clear log

Input age (required number)

Input pin (required digits length=4)

Input password (required)

Input password again (required compare=pass1)

Submit

```
<div class="row">
    <div class="cell-md-6">
        <form data-role="validator" action="javascript:"
              data-on-error="$('#handling-errors-onError').append(this.name + ': ' + this.value + '<br/>')">
            <div class="form-group">
                <label>Input name (<small>required</small>)</label>
                <input type="text" data-validate="required" name="nickname">
            </div>
            <div class="form-group">
                <label>Input age (<small>required number</small>)</label>
                <input type="text" data-validate="required number" name="age">
            </div>
            <div class="form-group">
                <label>Input pin (<small>required digits length=4</small>)</label>
                <input type="text" data-validate="required digits length=4" name="pin">
            </div>
            <div class="form-group">
                <label>Input password (<small>required</small>)</label>
                <input type="password" data-validate="required" name="pass1">
            </div>
            <div class="form-group">
                <label>Input password again (<small>required compare=pass1</small>)</label>
                <input type="password" data-validate="required compare=pass1" name="pass1_1">
                <span class="invalid_feedback">
```

```
            Pass1 and Pass2 must be equal.
        </span>
            </div>
            <br />
            <button class="button success">Submit</button>
        </form>
    </div>
    <div class="cell-md-6">
        <button class="button" onclick="$('#handling-errors-onError').html('')">Clear log</button>
        <div id="handling-errors-onError"></div>
    </div>
</div>
```

onErrorForm

Event `onErrorForm` generated for form when input not passed validation. Argument for this event is object: `{input: el, name = '...', value: ..., funcs: [...], errors: [...]}`.

input - html input element
name - input name
value - input value
funcs - input validation functions as array
errors - not passed validation functions as array

Input name (required)

Input age (required number)

Input pin (required digits length=4)

Input password (required)

Input password again (required compare=pass1)

Submit

```
<div class="row">
    <div class="cell-md-6">
        <form data-role="validator" action="javascript:"
              data-on-error="
              var log = arguments[0];
              var res = $('#handling-errors-onErrorForm');
              res.html('');
              $.each(log, function(){
                    res.append(this.name + ': ' + this.errors.join(',') + '')
              })
              ">
            <div class="form-group">
                <label>Input name (<small>required</small>)</label>
                <input type="text" data-validate="required" name="nickname">
            </div>
            <div class="form-group">
                <label>Input age (<small>required number</small>)</label>
                <input type="text" data-validate="required number" name="age">
            </div>
            <div class="form-group">
                <label>Input pin (<small>required digits length=4</small>)</label>
                <input type="text" data-validate="required digits length=4" name="pin">
            </div>
            <div class="form-group">
                <label>Input password (<small>required</small>)</label>
                <input type="password" data-validate="required" name="pass1">
            </div>
            <div class="form-group">
                <label>Input password again (<small>required compare=pass1</small>)</label>
                <input type="password" data-validate="required compare=pass1" name="pass1_1">
                <span class="invalid_feedback">
            Pass1 and Pass2 must be equal.
        </span>
            </div>
            <br />
            <button class="button success">Submit</button>
        </form>
    </div>
    <div class="cell-md-6">
        <button class="button" onclick="$('#handling-errors-onError').html('')">Clear log</button>
        <div id="handling-errors-onError"></div>
    </div>
</div>
```

Interactive check

Thanks to Prabakar Decipher

If you set attribute `data-interactive-check` to `true`, validator checks input on user change input value.

First name

Enter first name

```
<form data-role="validator"
        action="javascript:" data-interactive-check="true">
    <div class="row mb-2">
        <div class="cell-md-6">
            <label>First name</label>
```

```
            <input type="text" data-validate="minlength=6" placeholder="Enter first name">
            <span class="invalid_feedback">
                Input correct name with min length 6 symbols
            </span>
        </div>
    </div>
</form>
```

This option only check input. When form submitted, validator check all values again.

Validating without a form

Also, you can use **validator functions** without a form. For use it without form, in `Metro 4` present object `Metro.validator` with all validation functions.

```
var validator = Metro.validator;
var field = $("#input_id"); // for input must be defined validation functions

validator.color("red") // return true
validator.color("blueRed") // return false

validator.validate(field, null) // return true or false
```