Frontend form validation

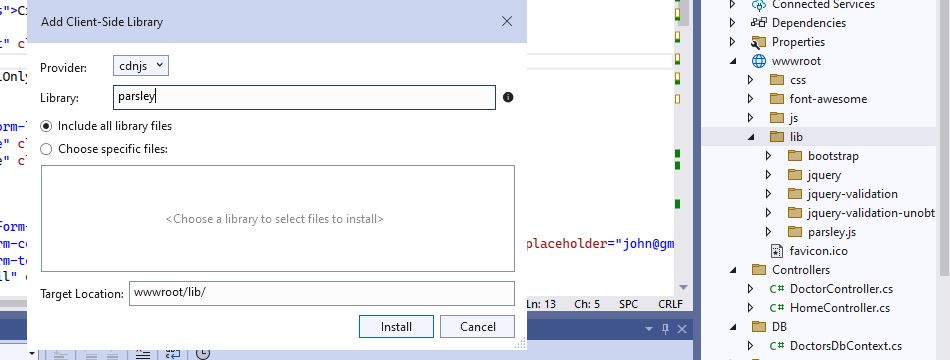
Parsley is a javascript form validation **library**. It helps you provide your users with feedback on their form submission before sending it to your server. It saves you bandwidth, server load and it saves time for your users.

Javascript form validation is not necessary, and if used, it **does not replace strong backend server validation.**

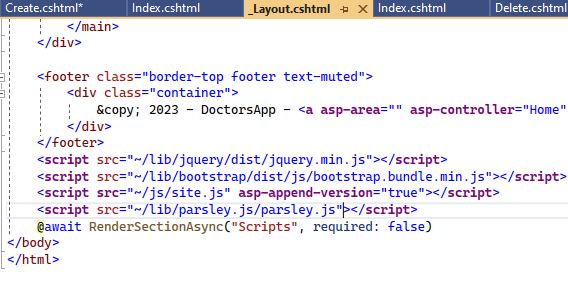
That's why Parsley is here: to let you define your general form validation, implement it on the backend side, and simply port it frontend-side, with maximum respect to user experience best practices.

***Parsley's current stable and supported versions are 2.x. If you still use a 1.x version***, [here is the related doc](http://parsleyjs.github.io/Parsley-1.x). But don't forget to [upgrade](https://github.com/guillaumepotier/Parsley.js/blob/master/UPGRADE-2.0.md)!

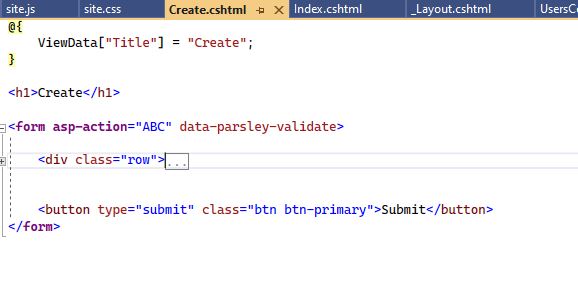
Here I am show how to add parsley validation in asp.net core mvc project:

First of all go to **> wwwroot > lib> add> client-sidelibrary> write parsley.js >Install** It in your project

***Then Go to Shared folder > \_Layout.cshtml > add this line***



<**script** src="~/lib/parsley.js/parsley.js"></**script**>

Now GoTo your Create/ Insert Form Page > Add validation>

Add “ data-parsley-validate ” First of your form.

Now Add text field Validation :



**Ex for name:** <input type="text" class="form-control" id="txtFirstName" placeholder="John" required>

**Ex for email:** <input type="email" class="form-control" id="txtEmail" placeholder="john@gmail.com" required>

**Ex for password:** <input type="password" class="form-control" id="txtPassword" placeholder="Password" data-parsley-minlength="8" data-parsley-uppercase="1" data-parsley-lowercase="1" data-parsley-special="1" data-parsley-number="1" data-parsley-error-message="Your password must 8 character long, contain at least (1) lowercase, (1) uppercase, (1) number and (1) special character." required>

**Ex for confirm-password:** <input type="password" class="form-control" id="txtConfirmPassword" placeholder="Password" data-parsley-equalto="#txtPassword" required>

***Form Ex:***

| **Property** | **Default** | **Description** |
| --- | --- | --- |
| data-parsley-namespace #2.0 | data-parsley- | Namespace used by Parsley DOM API to bind options from DOM. [See more](https://parsleyjs.org/doc/index.html#data-attrs) |
| data-parsley-validate #2.0 |  | Auto bind your form with Parsley validation on document load. |
| data-parsley-priority-enabled #2.0 | true | Either validate higher priority constraints first and stop on first failure (true), or validate all constraints simultaneously and show all the failing ones (false). |
| data-parsley-inputs #2.0 | input, textarea, select | When looking for fields within a form, Parsley uses this selector. The fields found will then be filtered using the excluded option below. |
| data-parsley-excluded #2.0 | input[type=button], input[type=submit], input[type=reset], input[type=hidden] | Form fields that won't be validated by Parsley. For example, if you want to add disabled and hidden fields to the existing list, use:  data-parsley-excluded="input[type=button], input[type=submit], input[type=reset], input[type=hidden], [disabled], :hidden" |

#### Methods ex:

| **Method** | **Returns** | **Description** |
| --- | --- | --- |
| whenValid({group, force}) #2.2 | promise | Returns a jQuery promise that will be fulfilled if and only if the Form is valid. **Does not affect UI nor fires**[**events**](https://parsleyjs.org/doc/index.html#events)**.** If group is given, it only validates fields that belong to this group. If force is given, it force validates even non-required fields ([See example](https://parsleyjs.org/doc/examples/events.html)) |
| isValid({group, force}) #2.0 | boolean or null | Similar to whenValid but returns true if the promise is already fulfilled, false if already rejected, and null if the validation is still pending. |
| whenValidate({group, force}) #2.0 | promise | Validate form. Prevents submission if not valid. **Fires**[**events**](https://parsleyjs.org/doc/index.html#events)**and affects UI.**. You can only validate a certain group of fields by specifying optional group string parameter. If group is given, it only validates fields that belong to this group. If force is given, it force validates even non required fields ([See example](https://parsleyjs.org/doc/examples/events.html)). Same result as whenValid. |
| validate({group, force}) #2.0 | boolean or null | Same as whenValidate except it returns true if the promise is already fulfilled, false if already rejected, and null if the validation is still pending. |
| refresh() #2.8 |  | Forces a refresh of the form and its field. Parsley always refreshes before validation, but this may be helpful for dynamic changes that need to be applied immediately (e.g. dynamically adding an input with a trigger, changing the `inputs` or `excluded` options, etc.). |
| reset() #2.0 |  | Reset UI for this form and for its fields. |

# Built-in validators

### Overview

A *validator* is a method to determine if a given *value* (or sometimes sets of values) is acceptable or not, given some *requirement* parameters.

Parsley comes with many builtin validators and provides tools to specify your own.

### Builtin validators list

| **Name** | **API** | **Description** |
| --- | --- | --- |
| Required #2.0 | |  |  | | --- | --- | | required | **HTML5** | | data-parsley-required |  | | data-parsley-required="true" |  | | data-parsley-required="false" |  | | Validates that a required field has been filled with a non blank value. If data-parsley-required="false", validator is deactivated and the field is not required. |
| Email #2.0 | |  |  | | --- | --- | | type="email" | **HTML5** | | data-parsley-type="email" |  | | Validates that a value is a valid email address. |
| Number #2.0 | data-parsley-type="number" | Validates that a value is a valid number according to the given step, min and original value. The default step for HTML5 is "1", which is so counterintuive that Parsley uses a default step of "any" for data-parsley-type="number". **Warning!** HTML5 type="number" can be counterintuitive. The default step of '1' is near useless. Moreover, for browsers that support it, the value accessible from javascript in case of invalid input is "", so you will never get an error (unless it is also required). |
| Integer #2.0 | |  |  | | --- | --- | | type="number" | **HTML5** | | data-parsley-type="integer" |  | | Validates that a value is a valid integer. |
| Digits #2.0 | data-parsley-type="digits" | Validates that a value is only digits. |
| Alphanum #2.0 | data-parsley-type="alphanum" | Validates that a value is a valid alphanumeric string. |
| Url #2.0 | |  |  | | --- | --- | | type="url" | **HTML5** | | data-parsley-type="url" |  | | Validates that a value is a valid url. |
| Minlength #2.0 | |  |  | | --- | --- | | minlength="6" | **HTML5** | | data-parsley-minlength="6" |  | | Validates that the length of a string is at least as long as the given limit. See also [length](https://parsleyjs.org/doc/index.html#psly-validator-length) |
| Maxlength #2.0 | |  |  | | --- | --- | | maxlength="10" | **HTML5** | | data-parsley-maxlength="10" |  | | Validates that the length of a string is not longer than the given limit. See also [length](https://parsleyjs.org/doc/index.html#psly-validator-length) |
| Length #2.0 | |  |  | | --- | --- | | minlength="6" maxlength="10" | **HTML5** | | data-parsley-length="[6, 10]" |  | | Validates that a given string length is between some minimum and maximum value. Specifying both HTML5 attributes minlength and maxlength will enable this validator, instead of both data-parsley-min and data-parsley-max validators. |
| Min #2.0 | |  |  | | --- | --- | | min="6" | **HTML5** | | data-parsley-min="6" |  | | Validates that a given input (number or date) or date is greater than or equal to some minimum (number or date.) |
| Max #2.0 | |  |  | | --- | --- | | max="10" | **HTML5** | | data-parsley-max="10" |  | | Validates that the given input (number or date) is less than or equal to some maximum value (number or date). |
| Range #2.0 | |  |  | | --- | --- | | type="range" | **HTML5** | | data-parsley-range="[6, 10]" |  | | Validates that a given value (number or date) is between some minimum and maximum values (numbers or dates). |
| Pattern #2.0 | |  |  | | --- | --- | | pattern="\d+" | **HTML5** | | data-parsley-pattern="\d+" |  | | Validates that a value matches a specific regular expression (regex). Note that patterns are anchored, i.e. must match the whole string. Parsley deviates from the standard for patterns looking like /*pattern*/*{flag}*; these are interpreted as literal regexp and are not anchored. |
| MinCheck #2.0 | data-parsley-mincheck="3" | Validates that a certain minimum number of checkboxes in a group are checked. |
| MaxCheck #2.0 | data-parsley-maxcheck="3" | Validates that a certain maximum number of checkboxes in a group are checked. |
| Check #2.0 | data-parsley-check="[1, 3]" | Validates that the number of checked checkboxes in a group is within a certain range. |
| Equalto #2.0 | data-parsley-equalto="#anotherfield" | Validates that the value is identical to another field's value (useful for password confirmation check). |