Lab 4 Part B: Form validator- Packages and libraries

The `FormValidator` can be used to validate the form elements before submitting to the server. That is, when typing wrong input values in the form element, it will show an alert message to notify the correction about the mistakes and it won't allow to submit the form until the input gets correct value.

index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>EJ2 Form Validator</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Typescript UI Controls" />
    <meta name="author" content="Syncfusion" />
    <link href="index.css" rel="stylesheet" />
    <link href="//cdn.syncfusion.com/ej2/ej2-base/styles/material.css"
rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scri
pt>
    <script src="systemjs.config.js"></script>
</head>

<body>
    <div id='loader'>Loading....</div>
    <div id='container'>
        <form id="form-element" class="form-horizontal">
            <div class="form-group">
                <label class="col-sm-3 control-label" for="user">User
Name</label>
                <div class="col-sm-6">
                    <input type="text" id="required" name="user" class="form-
control" />
                </div>
            </div>
            <div class="form-group">
                <label class="col-sm-3 control-label" for="age">Age</label>
                <div class="col-sm-6">
                    <input type="text" id="number" name="age" class="form-
control" />
                </div>
            </div>
        </form>
    </div>
```

```
</body>

</html>
```

index.ts

```
import {FormValidator, FormValidatorModel} from '@syncfusion/ej2-inputs';
let options: FormValidatorModel = {
    rules: {
        'user': { required: true },
        'age': { number: true, max: 25 }
    }
}
let formObject: FormValidator = new FormValidator('#form-element', options);
```

Validation Rules

Default Rules

The `FormValidator` has following default validation rules, which are used to validate the form.

Rules	Description	Example
`required`	The form input element must have any input values	a or 1 or -
`email`	The form input element must have valid `email` format values	`<form@syncfusion.com>`
`url`	The form input element must have valid `url` format values	http://syncfusion.com/
`date`	The form input element must have valid `date` format values	05/15/2017
`dateIso`	The form input element must have valid `dateISO` format values	2017-05-15
`number`	The form input element must have valid `number` format values	1.0 or 1
`digit`	The form input element must have valid `digit` values	1
`maxLength`	Input value must have less than or equal to `maxLength` character length	if `maxLength: 5`, **check** is valid and **checking** is invalid

Rules	Description	Example
`minLength`	Input value must have greater than or equal to `minLength` character length	if `minLength: 5`, **testing** is valid and **test** is invalid
`rangeLength`	Input value must have value between `rangeLength` character length	if `rangeLength: [4,5]`, **test** is valid and **key** is invalid
`range`	Input value must have value between `range` number	if `range: [4,5]`, **4** is valid and **6** is invalid
`max`	Input value must have less than or equal to `max` number	if `max: 3`, **3** is valid and **4** is invalid
`min`	Input value must have greater than or equal to `min` number	if `min: 4`, **5** is valid and **2** is invalid
`regex`	Input value must have valid `regex` format	if `regex: '^[A-z]+$'`, **a** is valid and **1** is invalid

The `rules` option should map the input element's `name` attribute. The `FormValidator` library only validates the mapped input elements.

Defining Custom Rules

You can also define custom rules in the `rules` property and validate the form with custom logics.

The custom validation method need to return the boolean value for validating an input.

index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>EJ2 Form Validator</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Typescript UI Controls" />
    <meta name="author" content="Syncfusion" />
    <link href="index.css" rel="stylesheet" />
    <link href="//cdn.syncfusion.com/ej2/ej2-base/styles/material.css"
rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scri
pt>
    <script src="systemjs.config.js"></script>
</head>
```

```
<body>
    <div id='loader'>Loading....</div>
    <div id='container'>
        <form id="form-element" class="form-horizontal">
            <div class="form-group">
                <label class="col-sm-3 control-label" for="user">User
Name</label>
                <div class="col-sm-6">
                    <input type="text" id="required" name="user" class="form-
control" />
                </div>
            </div>
            <div class="form-group">
                <label class="col-sm-3 control-label"
for="password">Password</label>
                <div class="col-sm-6">
                    <input type="password" id="number" name="password"
class="form-control" />
                </div>
            </div>
        </form>
    </div>
</body>

</html>
```

index.ts

```
import {FormValidator, FormValidatorModel} from '@syncfusion/ej2-inputs';

let customFn: (args: { [key: string]: string }) => boolean = (args: { [key:
string]: string }) => {
    return args['value'].length >= 5;
};
let options: FormValidatorModel = {
    rules: {
        'user': { required: true },
        'password': { minLength: [customFn, 'Need atleast 5 letters'] }
    }
}
let formObject: FormValidator = new FormValidator('#form-element', options);
```

Adding or Removing Rules

After creating `FormValidator` object, you can add more rules to an input element by using `addRules` method and you can also remove an existing rule from the input element by using `removeRules` method.

```
import {FormValidator, FormValidatorModel} from '@syncfusion/ej2-inputs';

let options: FormValidatorModel = {
    rules: {
        'user': { required: true },
        'age': { number: true }
    }
}
let formObject: FormValidator = new FormValidator('#form-element', options);
// Add email rule to user element
```

```
formObject.addRules('user', { maxLength: 7 });
// Remove number rule from age element
formObject.removeRules('age', ['number']);
```

Validating a Form

You can manually trigger validation by calling the `validate` method of `FormValidator`.
index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>EJ2 Form Validator</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Typescript UI Controls" />
    <meta name="author" content="Syncfusion" />
    <link href="index.css" rel="stylesheet" />
    <link href="//cdn.syncfusion.com/ej2/ej2-base/styles/material.css"
rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scri
pt>
    <script src="systemjs.config.js"></script>
</head>

<body>
    <div id='loader'>Loading....</div>
    <div id='container'>
        <form id="form-element" class="form-horizontal">
            <div class="form-group">
                <label class="col-sm-3 control-label"
for="product_name">Product Name</label>
                <div class="col-sm-6">
                    <input type="text" id="required" name="product_name"
class="form-control" />
                </div>
            </div>
            <div class="form-group">
                <label class="col-sm-3 control-label"
for="rating">Rating</label>
                <div class="col-sm-6">
                    <input type="text" id="number" name="rating" class="form-
control" />
                </div>
            </div>
        </form>
    </div>
</body>

</html>
```

index.ts

```
import {FormValidator, FormValidatorModel} from '@syncfusion/ej2-inputs';

let options: FormValidatorModel = {
    rules: {
        'product_name': { required: true },
        'rating': { range: [1,5] }
    }
}
let formObject: FormValidator = new FormValidator('#form-element', options);
// validate all input elements in the form
formObject.validate();
```

Validating a Single Element

The `validate` method have an optional argument, where you can pass an input element's name attribute to validate its value against the defined rule.

```
import {FormValidator, FormValidatorModel} from '@syncfusion/ej2-inputs';

let options: FormValidatorModel = {
    rules: {
        'user': { required: true },
        'age': { number: true }
    }
}
let formObject: FormValidator = new FormValidator('#form-element', options);
// validate user element alone
formObject.validate('user');
```

Error Messages

The `FormValidator` provides default error messages for all default validation rules. It is tabulated as follows

Rules	message
`required`	This field is required.
`email`	Please enter a valid email address.
`url`	Please enter a valid URL.
`date`	Please enter a valid date.
`dateIso`	Please enter a valid date (ISO).
`number`	Please enter a valid number.
`digit`	Please enter only digits.
`maxLength`	Please enter no more than {0} characters.

Rules	message
minLength	Please enter at least {0} characters.
rangeLength	Please enter a value between {0} and {1} characters long.
range	Please enter a value between {0} and {1}.
max	Please enter a value less than or equal to {0}.
min	Please enter a value greater than or equal to {0}.
regex	Please enter a correct value.

Customizing Error Messages

The default error message for a rule can be customizable by defining it along with concern rule object as follows.

index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>EJ2 Form Validator</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Typescript UI Controls" />
    <meta name="author" content="Syncfusion" />
    <link href="index.css" rel="stylesheet" />
    <link href="//cdn.syncfusion.com/ej2/ej2-base/styles/material.css"
rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scri
pt>
    <script src="systemjs.config.js"></script>
</head>

<body>
    <div id='loader'>Loading....</div>
    <div id='container'>
        <form id="form-element" class="form-horizontal">
            <div class="form-group">
                <label class="col-sm-3 control-label" for="user">User
Name</label>
                <div class="col-sm-6">
```

```
                    <input type="text" id="required" name="user" class="form-
control" />
                </div>
            </div>
            <div class="form-group">
                <label class="col-sm-3 control-label"
for="password">Password</label>
                <div class="col-sm-6">
                    <input type="password" id="number" name="password"
class="form-control" />
                </div>
            </div>
        </form>
    </div>
</body>

</html>
```

index.ts

```
import {FormValidator, FormValidatorModel} from '@syncfusion/ej2-inputs';

let options: FormValidatorModel = {
    rules: {
        'user': { required: true },
        'password': { minLength: [6, 'Need atleast 6 character length'] }
    }
}
let formObject: FormValidator = new FormValidator('#form-element', options);
```

Customizing Error Placement

The `FormValidator` has an event `customPlacement` which can be used to place the error message from default position to desired custom location.

index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>EJ2 Form Validator</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="Typescript UI Controls" />
    <meta name="author" content="Syncfusion" />
    <link href="index.css" rel="stylesheet" />
    <link href="//cdn.syncfusion.com/ej2/ej2-base/styles/material.css"
rel="stylesheet" />
    <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
rel="stylesheet" />
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/systemjs/0.19.38/system.js"></scri
pt>
```

```
    <script src="systemjs.config.js"></script>
</head>

<body>
    <div id='loader'>LOADING....</div>
    <div id='container'>
        <form id="form-element" class="form-horizontal">
            <div class="form-group">
                <label class="col-sm-3 control-label" for="user">User
Name</label>
                <div class="col-sm-6">
                    <input type="text" id="required" name="user" class="form-
control" />
                </div>
            </div>
            <div class="form-group">
                <label class="col-sm-3 control-label"
for="password">Password</label>
                <div class="col-sm-6">
                    <input type="password" id="number" name="password"
class="form-control" />
                </div>
            </div>
            <div class="form-group">
                <div class="col-sm-3 control-label"></div>
                <div class="col-sm-6">
                    <div id='error'></div>
                </div>
            </div>
        </form>
    </div>
</body>

</html>
```

index.ts

```
import {FormValidator, FormValidatorModel} from '@syncfusion/ej2-inputs';

let options: FormValidatorModel = {
    rules: {
        'user': { required: [true, 'User Name: required'] },
        'password': { minLength: [5, 'Password: need atleast 5 characters'] }
    },
    customPlacement: (inputElement: HTMLElement, error: HTMLElement)=>{
        document.getElementById('error').appendChild(error);
    }
}
let formObject: FormValidator = new FormValidator('#form-element', options);
```

Reference:

https://help.syncfusion.com/?_ga=2.172328291.1780888885.1614885106-1018006508.1614885106