HTML5 Validation

HTML5 Validation

When we submit a form, the data in the form will be sent to the server, before that we need to make sure that all the required details are filled out also in the correct format. The process of ensuring or validating the data before submitting to the server is called Client-side form validation.

Let us discuss form validation. For example, we submit any registration form, we may come across such messages listed below :

- "This field is required" (it can't be blank).
- "Please enter the valid phone number" (it should contain the only number).
- "Invalid email address" (it should be in "lmn@asd.com" format).
- "Your password must include one number, one uppercase letter, one lowercase letter, and one special character".

The browser displays the above messages by validating the data entered the registration form. It checks whether the data is in the correct format and satisfies the constraints set by the application or not. If the browser validates the data, then it is called as *client-side validation*. Validation done by the server is called as *server-side validation*.

There are two different types of client-side validation.

- **Built-in form validation** - It uses HTML5 form validation features.
- **JavaScript validation** - It is coded using JavaScript. This validation is completely customizable.

Built-in form validation

We have attributes that can be used with the form elements for validation. Some of the attributes are listed below:

- **required**: Used when the user must fill the field before submitting the form.
- **minlength and maxlength**: Used to specify the minimum and maximum length of the text.
- **min and max**: Used to specify the min and max values for the numerical fields.
- **type**: Defines the data should be a number or an email address or other predefined type.
- **pattern**: Defines a pattern (regular expression) the entered data needs to follow.

If the data entered in a form satisfies the constraints are considered **valid**. If not, it is considered **invalid**. We use `: valid` and `: invalid` CSS pseudo-class to differentiate between valid and invalid input fields. The `:invalid` CSS pseudo-class used to select and style form `<input>` elements whose value is *invalid* according to the validation attributes specified in the `<input>` element. Similarly, the `:valid` CSS pseudo-class selects and styles the vaild form input elements.

For example, the email inputs `(<input type="email">)` whose value does not match a valid email address pattern then the style defined by the `: invalid` CSS pseudo-class. Below, we have HTML and CSS code for this example.

```
HTML:

 <form>
     <label>
        <span>Email:</span>
        <input type="email" id="email" placeholder="name@domain.com">
    </label>
</form>

CSS:

//:valid
.valid {
    input[type=email]:valid { border-color: $g; }
}
//:invalid
.invalid {
    input[type=email]:invalid { border-color: $r; }
```