Client – Side Form Validation

Form Validation

Go to any popular site with a registration form, and you will notice that they provide feedback when you don't enter your data in the format they are expecting. You'll get messages such as:

* "This field is required" (You can't leave this field blank).
* "Please enter your phone number in the format xxx-xxxx" (A specific data format is required for it to be considered valid).
* "Please enter a valid email address" (the data you entered is not in the right format).
* "Your password needs to be between 8 and 30 characters long and contain one uppercase letter, one symbol, and a number." (A very specific data format is required for your data).

This is called form validation. When you enter data, the browser and/or the web server will check to see that the data is in the correct format and within the constraints set by the application. Validation done in the browser is called client-side validation, while validation done on the server is called server-side validation.

Using form Validation:

* Required
* Min and max length ( for string)
* Min an max (for number)
* Type and pattern

Psecudo class : valid let you applie style to valid elements

input:invalid:required {

background-image: linear-gradient(to right, pink, lightgreen);

}

The presence of the required attribute on any element that supports this attribute means the element matches the :required pseudoclass whether it has a value or not. If the <input> has no value, the input will match the :invalid pseudoclass.

Validation Form JavaScript

The constrain validation API consist of a set methods and properties of the form DOM validation.

Using Fetch

The Fetch API provides a JavaScript interface for accessing and manipulating parts of the HTTP pipeline, such as requests and responses. It also provides a global fetch() method that provides an easy, logical way to fetch resources asynchronously across the network.

Basic Fetch request

fetch('http://example.com/movies.json')

.then(response => response.json())

.then(data => console.log(data));

Here we are fetching a JSON file across the network and printing it to the console. The simplest use of fetch() takes one argument — the path to the resource you want to fetch — and returns a promise containing the response (a Response object).

This is just an HTTP response, not the actual JSON. To extract the JSON body content from the response, we use the json() method (defined on the Body mixin, which is implemented by both the Request and Response objects.)