



MENU



All Tutorials

Java

Maven

Gradle

Servlet/Jsp

Thymeleaf

Spring

Struts2

Hibernate

Java Web Service

JavaFX

SWT

Oracle ADF

Android

iOS

Python

Swift

C#

C/C++

Ruby

Dart




Batch	
Database	
Oracle APEX	
Report	
Client	
ECMAScript / Javascript	
TypeScript	
NodeJS	
ReactJS	
Flutter	
AngularJS	
HTML	
CSS	
Bootstrap	
OS	
Git	
SAP	
Amazon AWS	
Others	

Javascript Form Validation Tutorial with Examples

View more Tutorials:

[ECMAScript, Javascript Tutorials](#)

1. [Form Validation](#)
2. [Simple example](#)
3. [Access the form data](#)
4. [Submit through Javascript](#)
5. [Validate automatically](#)

 Follow us on our fanpages to receive notifications every time there are new articles.  Facebook  Twitter

1- Form Validation

Quite regularly, you meet a **website** where users enter information into a form before sending it to the server, for example, the account registration form. The information that the user enters into the form needs to be validated to ensure data rationality.

Register

Create your account. It's free and only takes a minute.

First Name Last Name

Email

Password

Confirm Password

☐ I accept the [Terms of Use & Privacy Policy](#).

Register Now

Already have an account? [Sign in](#)

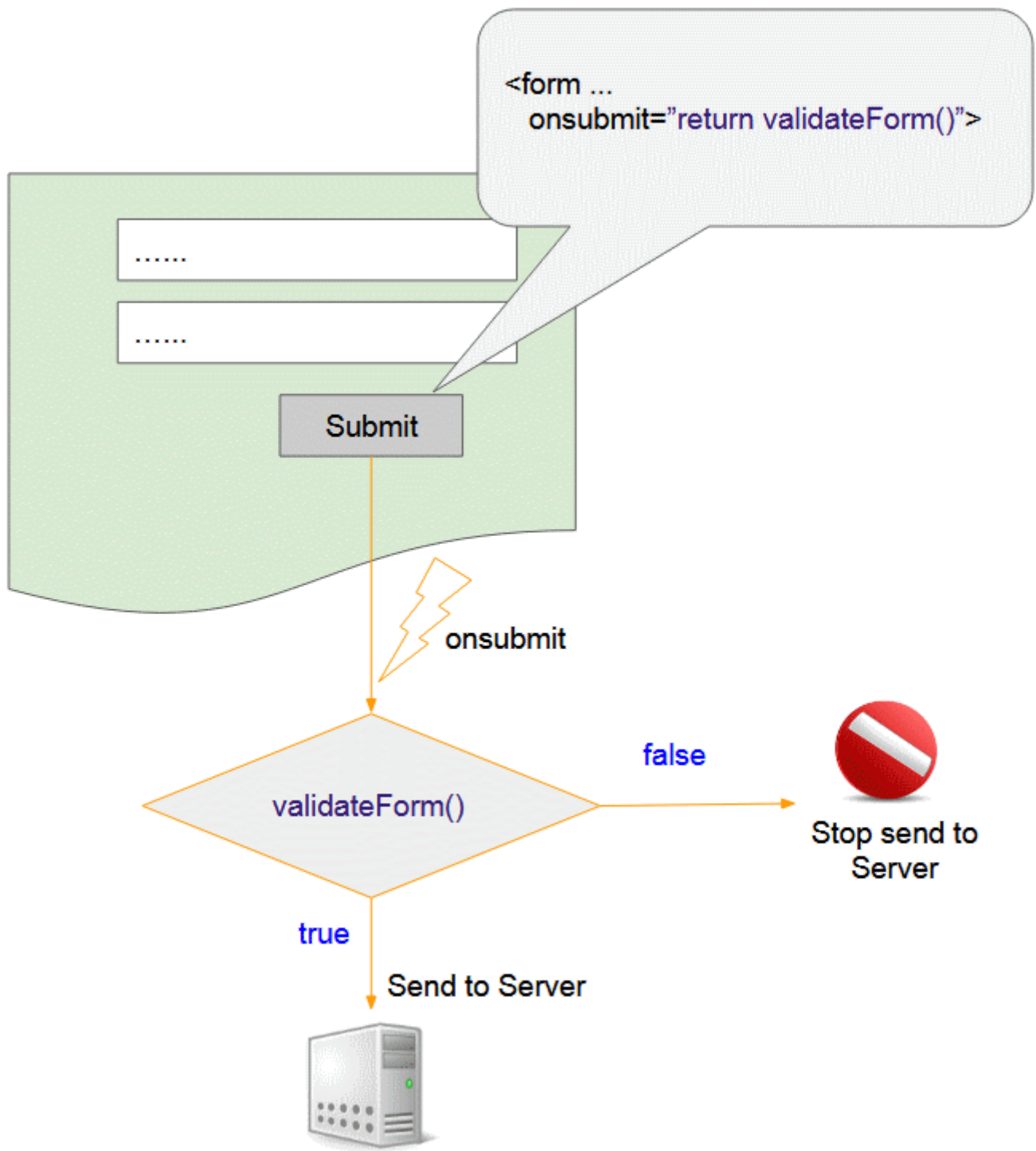
Some examples of authentication:

- Check to ensure that the data is not empty.
- Check email format
- Check telephone number format
- ..

There are basically 3 ways for data validation:

1. **form** data will be sent to the **server**, and validation will be done on the server side.
2. **form** data will be validated on the **client** side by using **Javascript**, which helps **server** not have to work too much and increase performance for the application.
3. Use both above methods to validate **form**.

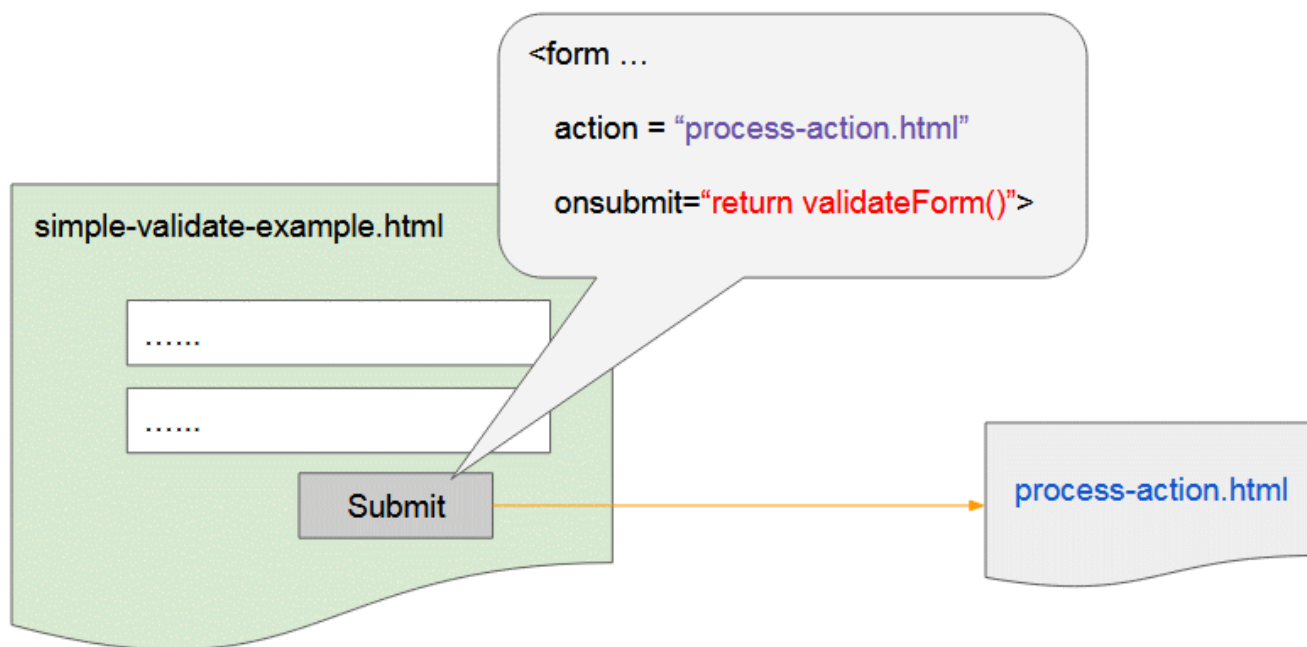
In this lesson, I will discuss using **Javascript** to validate **form**. Below is the illustration of the program's behavior when the user clicks the **Submit** button.



1. You have to register a function in combination with the **onsubmit** event of **form**. The duty of this function is to check the data which an user has entered in **form**, and return **true** if all the information entered by the user is valid and vice versa return **false**.
2. When the user clicks **Submit**, the function in combination with the **onsubmit** event will be called.
3. If the function in combination with the **onsubmit** event returns **true**, the data of **form** will be sent to the **server** and vice versa the **Submit** action will be cancelled.

2- Simple example

OK, this is a simple example helping you understand the operating rules of **Form** before practising more complex examples.



The **action** attribute of **<form>** is used to specify the page to which data will be given or in other words, this is the page that will process the data sent from the **<form>** of the current page.

//

*The pages processing the data sent from **form** are usually written by **Servlet/JSP, PHP** technology or a technology on the **Server** side instead of an **HTML** page. However, I do not mention data processing in this lesson.*

simple-validation-example.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello Javascript</title>
    <script type = "text/javascript">
      function validateForm() {
        var u = document.getElementById("username").value;
        var p = document.getElementById("password").value;

        if(u== "") {
          alert("Please enter your Username");
          return false;
        }
        if(p == "") {
          alert("Please enter you Password");
          return false;
        }
      }
    </script>
  </head>
  <body>
    <input type="text" value="username" />
    <input type="password" value="password" />
    <input type="submit" value="Submit" />
  </body>
</html>
```

```
}

    alert("All datas are valid!, send it to the server!")

    return true;
}
</script>
</head>
<body>

<h2>Enter your Username and Password</h2>

<div style="border:1px solid #ddd; padding: 5px;">
    <form method="GET" action="process-action.html" onsubmit = "return validateForm()">
        Username: <input type="text" name="username" id="username"/>
        <br><br>
        Password: <input type="password" name = "password" id="password"/>
        <br><br>
        <button type="submit">Submit</button>
    </form>
</div>

</body>
</html>
```

process-action.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Process Action</title>

</head>
<body>

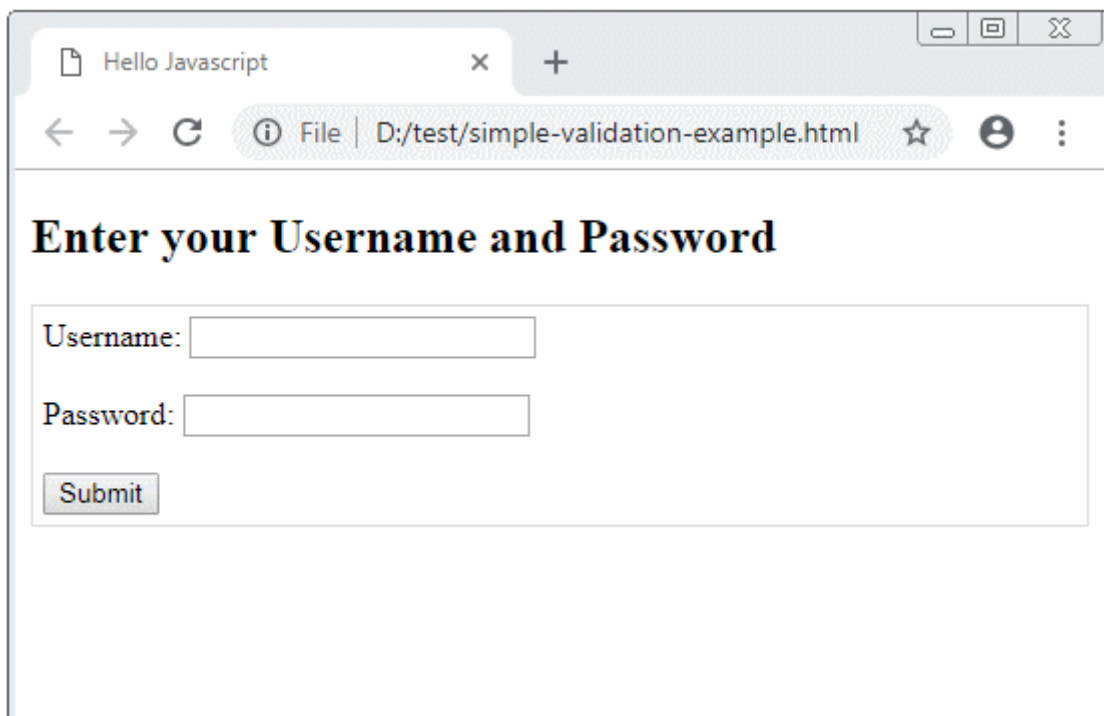
    <h3>Process Action Page</h3>

    OK, I got data!

    <br/><br/>

    <a href="javascript:history.back();">[Go Back]</a>

</body>
</html>
```



The screenshot shows a web browser window with a single tab titled 'Hello Javascript'. The address bar shows the file path 'D:/test/simple-validation-example.html'. The page content features a heading 'Enter your Username and Password' in a large, bold, black serif font. Below the heading is a form with a light gray border. Inside the form, there are two labels: 'Username:' and 'Password:', each followed by a text input field. At the bottom left of the form is a 'Submit' button with a gray gradient and a black border.

3- Access the form data

Access a field data through the field **ID**.

```
<input type="text" id="username"/>
<input type="password" id="password"/>
```

```
// Access field via ID:
var field = document.getElementById("fieldId");

var value = field.value;
```

Access **Form** fields through the **name** attribute:

```
<form name="myForm" ...>
  <input type="text" name="username"/>
  <input type="password" name = "password"/>
  <button type="submit">Submit</button>
</form>
```

```
// Get form via form name:
var myForm = document.forms["myForm"];

var u = myForm["username"].value;
var p = myForm["password"].value;
```

When a user enters inaccurate data on a **form** field, you should notify the user and **focus** on that field.

validation-example1.html

```
<!DOCTYPE html>

<html>
  <head>
    <title>Validation</title>
    <script type = "text/javascript">

      function validateForm() {
        // Get form via form name:
        var myForm = document.forms["myForm"];

        var u = myForm["username"].value;
        var p = myForm["password"].value;

        if(u== "") {
          alert("Please enter your Username");
          myForm["username"].focus(); // Focus
          return false;
        }
        if(p == "") {
          alert("Please enter you Password");
          myForm["password"].focus(); // Focus
          return false;
        }

        alert("All datas are valid!, send it to the server!")

        return true;
      }
    </script>
  </head>
  <body>

    <h2>Enter your Username and Password</h2>

    <div style="border:1px solid #ddd; padding: 5px;">

      <form name="myForm" method="GET" action="process-action.html" onsubmit = "return validateForm()">
        Username: <input type="text" name="username"/>
        <br><br>
        Password: <input type="password" name = "password"/>
        <br><br>
        <button type="submit">Submit</button>
      </form>

    </div>
```

```
</body>
</html>
```

Example: Ask an user to enter a number between 0 and 10.

validation-number-example.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Validation</title>
    <script type = "text/javascript">

      function validateForm() {

        var myField = document.getElementById("myNumber");
        var value = myField.value;

        if( value == "" || isNaN(value) || value < 0 || value > 10) {
          alert("Invalid input!");
          myField.focus();
          return false;
        }

        return true;
      }
    </script>
  </head>
  <body>

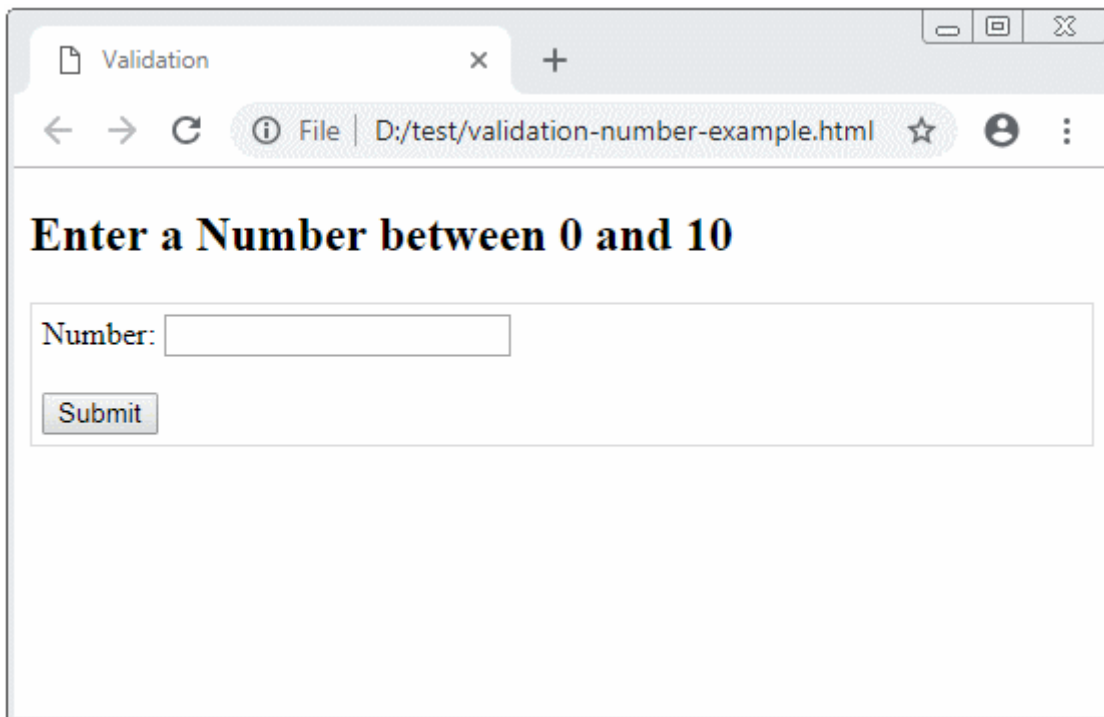
    <h2>Enter a Number between 0 and 10</h2>

    <div style="border:1px solid #ddd; padding: 5px;">

      <form name="myForm" action="process-action.html" onsubmit = "return validateForm()">
        Number: <input type="text" id= "myNumber"/>
        <br/><br/>
        <button type="submit">Submit</button>
      </form>

    </div>

  </body>
</html>
```



The screenshot shows a web browser window with a single tab titled 'Validation'. The address bar shows the file path 'D:/test/validation-number-example.html'. The main content area displays a form with the heading 'Enter a Number between 0 and 10'. Below the heading is a text input field with the label 'Number:' and a 'Submit' button.

4- Submit through Javascript

Clicking `<button type="submit">` or `<input type="submit">` inside **form** helps you to send the data of this **form** to the server, however, you can also do it through **Javascript**.

javascript-submit-example.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Javascript Submit</title>
    <script type = "text/javascript">

      function validateForm() {
        var name = document.forms["myForm"]["fullName"].value;
        if(name == "") {
          alert("Please enter your name");
          return false;
        }
        return true;
      }

      function submitByJavascript() {

        var valid = validateForm();
        if(!valid) {
          return;
        }

        var myForm = document.forms["myForm"];
```

```
        myForm.submit();
    }
</script>
</head>
<body>

<h2>Submit a form with Javascript</h2>

<div style="border:1px solid #ddd; padding: 5px;">

    <form name="myForm" action="process-action.html" onsubmit = "return validateForm()">
        Your Name: <input type="text" name = "fullName" value = ""/>
        <br/><br/>
        <button type="submit">Submit</button>
    </form>

</div>
<br/>

<!-- A Button outside the form -->
Button outside the form:
<button onclick="submitByJavascript()">Click Me to submit form</button>

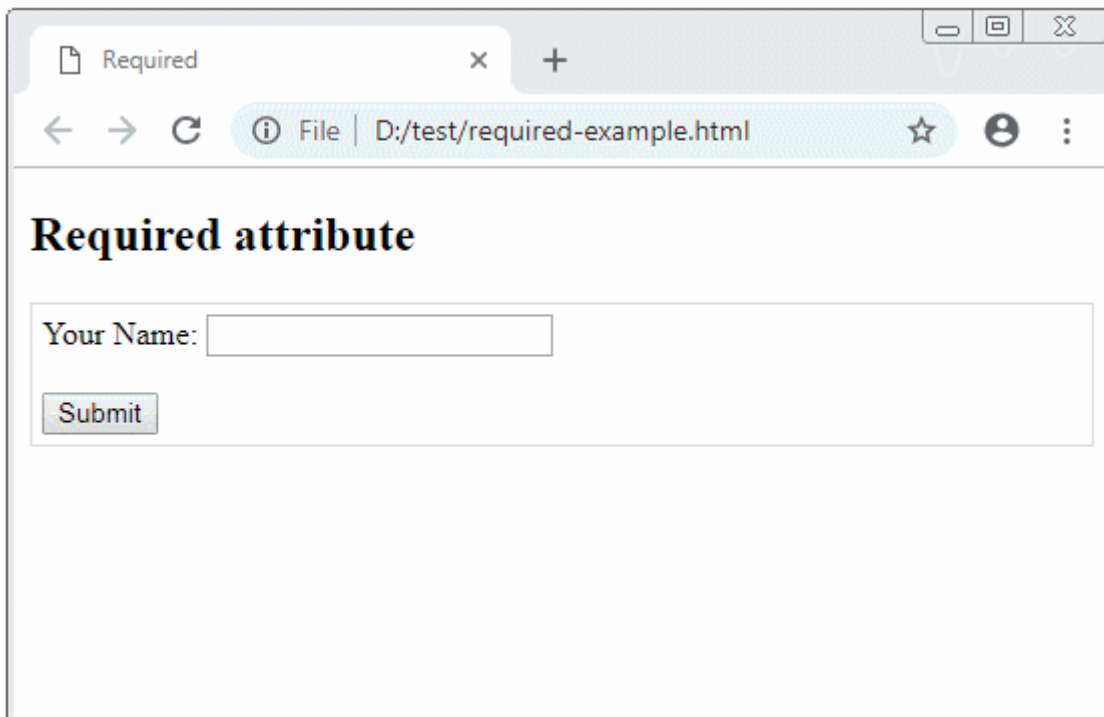
</body>
</html>
```

5- Validate automatically

The browser can automatically validate several types of data on the **form**, such as adding a **required** attribute to a form field to tell the browser that this field is mandatory. The browser will automatically check and notify an user if an user does not enter that field.

/

*Note: Too old browsers such as **IE** version 9 or older do not support automatic **validate**.*



required-example.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Required</title>

  </head>
  <body>

    <h2>Required attribute</h2>

    <div style="border:1px solid #ddd; padding: 5px;">

      <form name="myForm" action="process-action.html" onsubmit = "return validateForm()">
        Your Name: <input type="text" name = "fullName" value = "" required/>
        <br/><br/>
        <button type="submit">Submit</button>
      </form>

    </div>

  </body>
</html>
```

Some `<input>` elements newly introduced in **HTML 5**, for example *color, date, datetime-local, email, month, number, range, search, tel, time, url, week*. These elements have special attributes to help the browser to know how to **validate** its data automatically. Below is list of some such attributes:

Attribute	Description
disabled	Specifies that the Input element should be disabled
max	Specifies the maximum value of an Input element
min	Specifies the minimum value of an Input element
pattern	Specifies the value pattern of an Input element
required	Specifies that the Input field requires an element
type	Specifies the type of an Input element

/

See the details of the list of **<input>** elements and attributes corresponding to each of these elements:

- [TODO Link?](#)

Example: A **<input type="number">** with **min**, **max** attributes, the browser will notify an user if he/she enters a number beyond the allowed range.

attr-min-max-example.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Min Max Attributes</title>

  </head>
  <body>

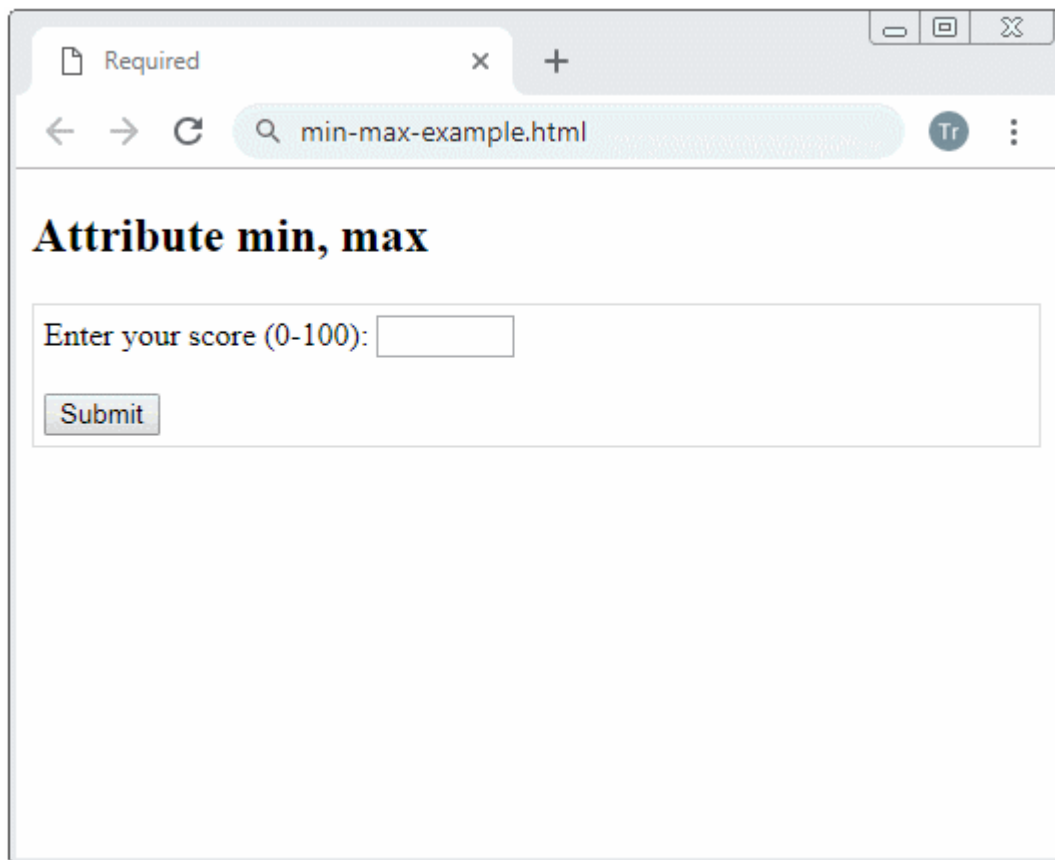
    <h2>Attribute min, max</h2>

    <div style="border:1px solid #ddd; padding: 5px;">

      <form name="myForm" action="process-action.html">
        Enter your score (0-100):
        <input type="number" name = "score" min= "0" max = "100"/>
        <br/><br/>
        <button type="submit">Submit</button>
      </form>

    </div>

  </body>
</html>
```



The screenshot shows a web browser window with a single tab titled 'Required'. The address bar shows 'min-max-example.html'. The page content includes a heading 'Attribute min, max' and a form with a text input field labeled 'Enter your score (0-100):' and a 'Submit' button.

Example: Require an user to enter a country code with 2 characters.

attr-pattern-example.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>pattern attribute</title>

  </head>
  <body>

    <h2>Attribute: pattern</h2>

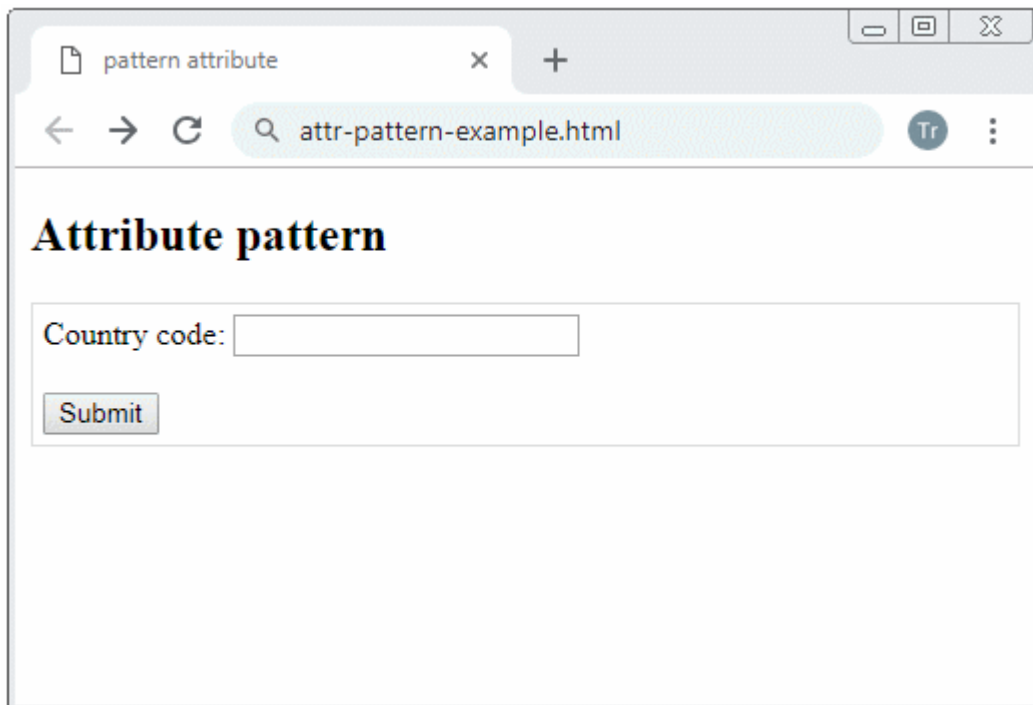
    <div style="border:1px solid #ddd; padding: 5px;">

      <form name="myForm" action="process-action.html">
        Country code:
        <input type="text" name = "countryCode" pattern="[A-Za-z]{2}"
          title="Two letter country code"/>
        <br/><br/>
        <button type="submit">Submit</button>
      </form>

    </div>

  </body>
```

```
</html>
```



Example: Ask an user to enter a password having at least 8 characters.

attr-pattern-example2.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>pattern attribute</title>

  </head>
  <body>

    <h2>Attribute: pattern</h2>

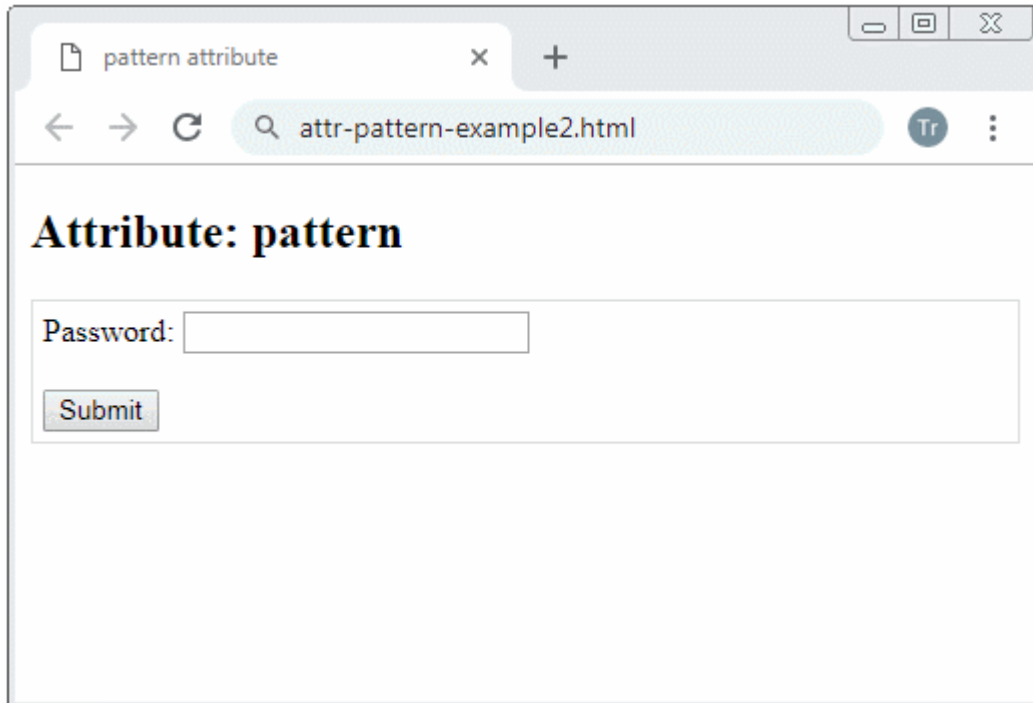
    <div style="border:1px solid #ddd; padding: 5px;">

      <form name="myForm" action="process-action.html">
        Password:
        <input type="password" name = "password" pattern=".{8,}"
          title="8 or more characters"/>
        <br/><br/>
        <button type="submit">Submit</button>
      </form>

    </div>

  </body>
```


</html>



Example: Ask an user to enter a strong password, having at least 8 characters, at least one uppercase, and at least one lowercase.

attr-pattern-password-example.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>pattern attribute</title>

  </head>
  <body>

    <h2>Attribute: pattern</h2>
    Password must contain 8 or more characters that are of at least one number,
    and one uppercase and lowercase letter:
    <br/><br/>

    <div style="border:1px solid #ddd; padding: 5px;">

      <form name="myForm" action="process-action.html">
        Password:
        <input type="password" name = "password"
          pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,}"
          title="Invalid password!"/>
        <br/><br/>
        <button type="submit">Submit</button>
      </form>
```

```
</div>

</body>
</html>
```

Example: Ask an user to enter an email address, use the **pattern** attribute to ensure that the user enters an email in the correct format.

attr-pattern-email-example.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>pattern attribute</title>

  </head>
  <body>

    <h2>Attribute: pattern</h2>

    <div style="border:1px solid #ddd; padding: 5px;">

      <form name="myForm" action="process-action.html">
        Email:
        <input type="password" name = "password"
          pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$"
          title="Invalid password!"/>
        <br/><br/>
        <button type="submit">Submit</button>
      </form>

    </div>






















































  </body>
</html>
```

View more Tutorials:

ECMAScript, Javascript Tutorials

Maybe you are interested

These are online courses outside the o7planning website that we introduced, which may include free or discounted courses.

-   The Complete JavaScript Bootcamp
-   Getting really good at JavaScript and TypeScript
-   Essentials of JavaScript Practice Coding Exercises Tips
-   Learn JavaScript From Scratch: Become Top Rated Web Developer
-   JavaScript For Beginners - Learn JavaScript From Scratch
-   Javascript for Beginners
-   Learning ECMAScript 6: Moving to the New JavaScript
-   JavaScript Intro to learning JavaScript web programming
-   JavaScript Dynamic Quiz Application from Scratch JSON AJAX
-   Byte-Sized-Chunks: Dynamic Prototypes in Javascript
-   Learn ECMAScript 2015 - ES6 Best Course
-   Getting started with javascript and its core concepts
-   Learn JavaScript Fundamentals
-   HTML CSS JavaScript: Most popular ways to code HTML CSS JS
-   *  * Master ECMAScript 2015 (ES6)
-   Quick JavaScript Core learning Course JavaScript Essentials
-   *  * Start 3D GIS Web Development in JavaScript
-   Learning JavaScript Programming Tutorial. A Definitive Guide
-   2D Game Development With HTML5 Canvas, JS - Tic Tac Toe Game
-   JavaScript for Beginning Web Developers
-   The Web Development Course: HTML5, CSS3, JavaScript
-   JavaScript in 55 Minutes
-   The complete beginner JavaScript ES5, ES6 and JQuery Course
-   JavaScript in Action JavaScript Projects
-   *  * Introductory To JavaScript - Learn The Basics of JavaScript

o7planning.org

Fanpages

[Facebook](#)[Twitter](#)

Websites

o7planning.orgdevstory.netcodestory.debetacode.netopenplanning.net

About Us

The website was created in March 2014 by a group of programmers and authors from Vietnam. Currently, the project supports 5 languages, including English, French, German, Russian and Vietnamese.

DMCA PROTECTED