Form Validation

Can be done in 2 ways

- **Built-in form validation** uses HTML5 form validation features. This validation generally doesn't require much JavaScript. Built-in form validation has better performance than JavaScript, but it is not as customizable as JavaScript validation.

- **JavaScript** validation is coded using JavaScript. This validation is completely customizable, but you need to create it all (or use a library).

Using built-in form validation

- This is done by using HTML 5 validation attributes on form elements.
- required: Specifies whether a form field needs to be filled in before the form can be submitted.
- minlength and maxlength: Specifies the minimum and maximum length of textual data (strings)
- min and max: Specifies the minimum and maximum values of numerical input types
- type: Specifies whether the data needs to be a number, an email address, or some other specific preset type.
- pattern: Specifies a regular expression that defines a pattern the entered data needs to follow.

- When an element is valid, the following things are true with CSS3 Pseudo selectors:

- The element matches the :valid CSS pseudo-class, which lets you apply a specific style to valid elements.
- If the user tries to send the data, the browser will submit the form, provided there is nothing else stopping it from doing so (e.g., JavaScript).

- When an element is invalid, the following things are true:

- The element matches the :invalid CSS pseudo-class, and sometimes other UI pseudo-classes (e.g., :out-of-range) depending on the error, which lets you apply a specific style to invalid elements.
- If the user tries to send the data, the browser will block the form and display an error message.

Example

```
<body>
   <form>

      <label for="myemail">Enter Email</label>

<input type="email" id="myemail"
name="myemail" required>

 <button>Submit</button>

   </form>
</body>
```

```
<head>
   <meta charset="utf-8">
   <title>Favorite fruit with required attribute</title>
   <style>


     input[type="email"]:invalid
     {
       border: 2px dashed red;
       background-image: linear-gradient(to right, yellow, lightgreen);
     }
     input:valid {
       border: 2px solid black;

     }
     input[type="email"]:focus
     {
        background-image: linear-gradient(to right, pink, lightgreen);
     }


   </style>
</head>
```

```
<body>
   <form>

  <label for="myno">Enter Age</label>

  <input type="number" id="myno" name="myno"
min="17" max="25" required>

<br/>

<label for="mylastname">Enter LastName</label>

 <input type="text" id="mylastname"
name="mylastname" />

 <button>Submit</button>

</form>
</body>
```

```
input[type="number"]:invalid
{
            box-shadow: 0 0 5px 1px red;
}

input[type="number"]:focus:invalid
 {
  box-shadow: none;
 }
  input:optional
 {
            border-color: grey;
   }
```

With Javascript

- With using The Constraint Validation API
- Without using API

Some errors that will not allow form to be submitted

- patternMismatch
- rangeOverflow or rangeUnderflow
- stepMismatch
- tooLong or tooShort
- typeMismatch
- valueMissing
- customError.

Example

<input name="tel2" type="tel" pattern="[0-9]{3}" placeholder="###" aria-label="3-digit prefix" size="2"/>	**input:invalid { border: red solid 3px; }**
<input type="number" min="20" max="40" step="2"/>	input:out-of-range { background-color: rgba(255, 0, 0, 0.25); }
<input type="email" value="example.com"/>	input:invalid { border: red solid 3px; }

Example for custom error validation.

```
<form>

<label for="mail">I would like you to provide me with an e-mail address:</label>

 <input type="email" id="mail" name="mail">

<button>Submit</button>

</form>
```

Note:
Validity Returns a **ValidityState** object that contains several properties describing the validity state of the element.
setCustomValidity Adds a custom error message to the element;

```
const email = document.getElementById("mail");

email.addEventListener("input", function (event)
{

if (email.validity.typeMismatch)
{

email.setCustomValidity("I am expecting an e-mail address!");
}

else { email.setCustomValidity(""); }
});
```

Validating forms without a built-in API

```
<script>
function validateform(){
var name=document.myform.name.value;
var
password=document.getElementById("pass").v
alue;

if (name==null || name==""){
  alert("Name can't be blank");
  return false;
}else if(password.length<6){
  alert("Password must be at least 6 characters
long.");
  return false;
  }
}
</script>
```

```
<body>
<form name="myform" method="post"
onsubmit="validateform()" >

Name: <input type="text"
name="name"><br/>

Password: <input type="password"
name="password" id="pass"><br/>

<input type="submit" value="Submit">
</form>
```