Hands-On Lab: Validating a JavaScript form

Skills Network

Estimated time 25 minutes

JavaScript is a client-side scripting language and is commonly used to create dynamic web pages. It helps you change web page content dynamically, as well as enables you to validate forms and perform other actions. In this lab, you will create an HTML form that uses JavaScript to validate input.

Objectives

After completing this lab, you will be able to:

1. Create a basic web form
2. Use the `<script>` tag
3. Add a JavaScript function
4. Access the form controls from JavaScript
5. Access a textbox and check if it is blank
6. Execute a set of statements based on a condition
7. Display error messages
8. Execute a function when the form is submitted
9. Practice Exercise: Add Javascript interactivity to the `Hands-on Lab: Unit Conversion using HTML5 Structural Elements` (completed earlier)

1. Create an HTML form

In this exercise, you will create a simple form that accepts a person's name and email address and then performs a simple validation on the entered input.

Go to the project explorer, clicking on **New File** symbol. A **New File** window opens. Enter `form_validation.html` as the file name and click **OK**. You are now ready to start creating the new form.

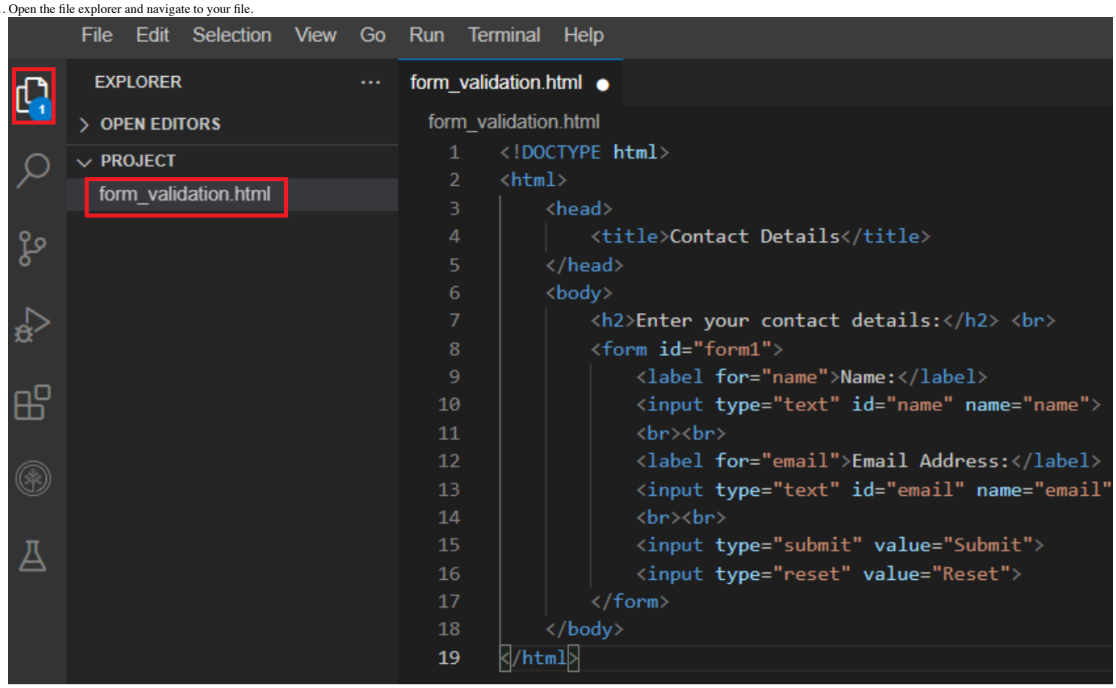
Let's start by creating a simple form designed to accept the user's name and email address. The form will have a **Submit** button and a **Reset** button.

Copy and paste the following code into your file to create the initial form without validation:

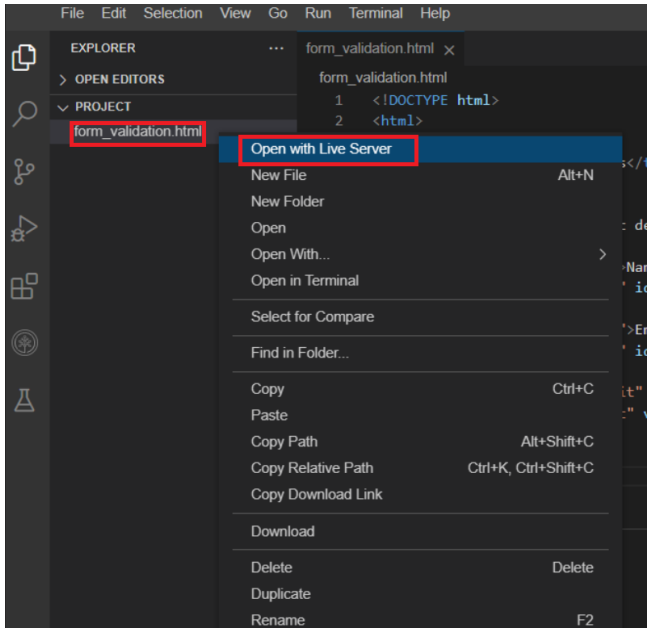
```
<!DOCTYPE html>
<html>
    <head>
        <title>Contact Details</title>
    </head>
    <body>
        <h2>Enter your contact details:</h2> <br>
        <form id="form1">
            <label for="name">Name:</label>
            <input type="text" id="name" name="name">
            <br><br>
            <label for="email">Email Address:</label>
            <input type="text" id="email" name="email">
            <br><br>
            <input type="submit" value="Submit">
            <input type="reset" value="Reset">
        </form>
    </body>
</html>
```

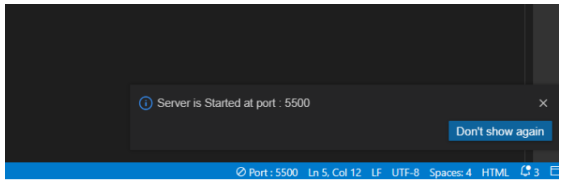
When you have pasted the code, save your file. To see how your HTML page will be displayed, you can use the built-in Live Server extension by following the instructions below.

1. Open the file explorer and navigate to your file.

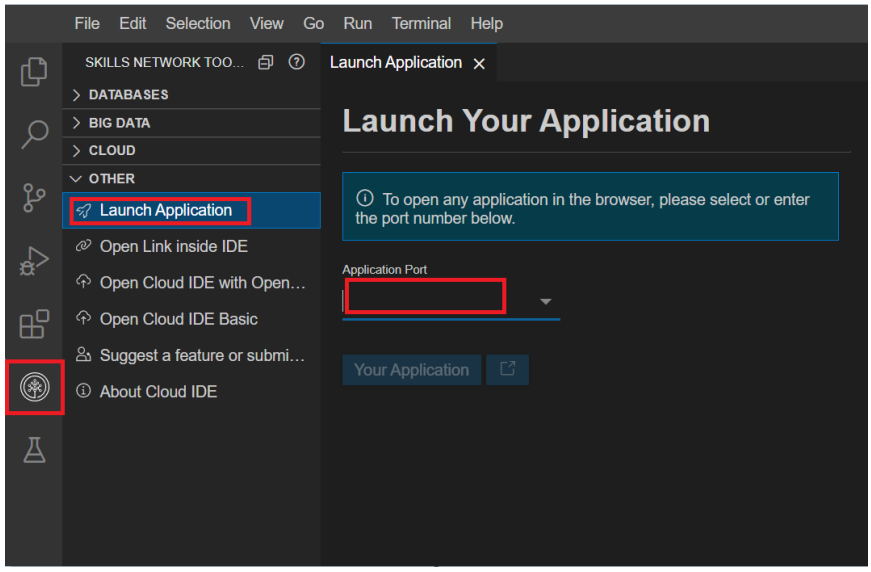


2. Right click on your file & click on 'Open with Live Server'

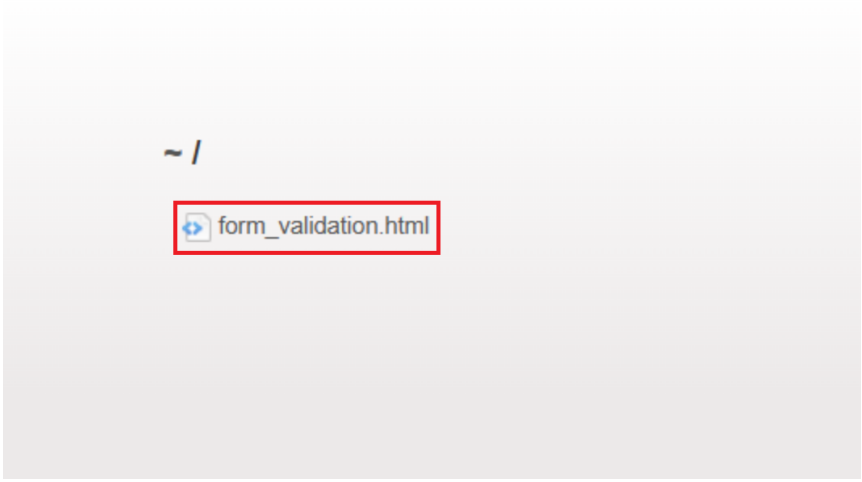




4.Click on the Skills Network button on the left, it will open the "Skills Network Toolbox". Then click the Other then Launch Application. From there you enter the port no. as 5500 and launch.



5.Click on the file name to view its preview.



After following the above steps, Your page should look like this:

Enter your contact details:

Name:

Email Address:

Submit Reset

2. Use the `<script>` tag

We use the `<script>` tag to embed executable code, usually JavaScript, into an HTML page. The tag can contain scripting statements, or it can refer to an external script file. We use a `type` attribute to specify the scripting language.

Although you can put the `<script>` tag anywhere in your HTML document, for this lab you'll put it in the `<head>` section.

Replace the `<head>` section of your file with the following code. It tells the browser that the code we are about to put inside the `<script>` tag must be executed as JavaScript.

```
<head>
    <script type="application/javascript">
    </script>
    <title>Contact Details</title>
</head>
```

3. Add a JavaScript function

Now you'll specify what happens when a user clicks the **Submit** button. We specify this behavior with a user-defined JavaScript *function*, which is a block of code that is executed when it's called. A function can be called any number of times.

A function in JavaScript looks like this:

```
function functon_name()
    {
        // code goes here
    }
```

Let's add an empty function that has the name `checkdata`. We will use this to validate the data in the form. Replace the `<script>` tags in your file with the following code:

```
<script type="application/javascript">
    function checkdata()
        {
        }
 </script>
```

4. Access HTML controls within JavaScript

The function you've created is intended to validate the contents of each of the input elements in the form. To access the data for an element, the script needs to refer to the correct element.

One way to identify an element is to use a method called `getElementById(elementID)`. The following line of code returns the element with the ID `name`:

```
document.getElementById("name");
```

The following lines of code enable you to access the `name` and `email` input elements of the form. Note that the id's of both these elements have already been specified in the HTML code. We will store the references to the elements in two JavaScript variables named `username` and `emailid`.

```
var username = document.getElementById("name");
var emailid = document.getElementById("email");
```

Copy the above code and paste it into your `checkdata()` function.

5. Access and check data

When the references to the elements are stored in the variables, the values of the elements can be retrieved using the `value` attribute. If `username` is the variable that contains the input element's reference, then its value can be accessed using

```
username.value
```

To check if this value is blank, we can use the following statement:

```
username.value == ""
```

"" indicates an empty string.

6. Execute a set of statements based on a condition

If the value is blank, we will print an error message and return the focus back to the empty element.

To perform this action, we use a JavaScript *conditional statement* called the `if` statement. The `if` conditional statement allows us to specify a block of code to be executed *if* a condition is true.

The syntax of the statement is as follows:

```
if(condition){
    //block of code to be executed, if the condition is true.
}
```

Let's check if the `username` value is empty by using an `if` statement. Copy this code and paste it at the bottom of your `checkdata()` function:

```
if(username.value==""){
    return false;
}
```

If the value is blank, the `return false;` statement returns a boolean value `false` from the `checkdata()` function that we added in step 3.

We will check all input elements of the form in this way to determine whether they are empty.

7. Display error messages

You can display a message to a user with the help of a pop-up alert message box. To do this, you will use the `alert` method.

Let's use this method within the function to alert the user.

```
if(username.value==""){
    alert("Please enter the name");
    username.focus();
    return false;
}
```

The `username.focus()` statement is used to bring the input focus back to the element where we found a problem, in this case, *username*.

Replace the current conditional in your function with this new code. Below, try to add the same conditional logic, but for the `email_address` element instead.

Next, we will indicate that none of the elements are blank by returning `true`. So, we need to add a `return true` statement at the end of the function.

It's a good practice to include comments in your code. Comments will help you and other programmers easily debug any errors that we might encounter while running the code. In JavasScript, we add single-line comments using two forward slashes: `//`

Our final `checkdata()` function with comments added looks like:

```
function checkdata(){
      //Create references to the input elements we wish to validate
      var username = document.getElementById("name");
      var email_address = document.getElementById("email");
      //Check if username field is empty
      if(username.value == ""){
            alert("Please enter the name");
            username.focus();
            return false;
      }
     //Check if email field is empty
      if(email_address.value == ""){
            alert("Please enter the email");
            email_address.focus();
            return false;
      }
     //If all is well return true.
      return true;
}
```

This function should now work as expected and is now ready to be called, so we can use it wherever we need.

8. Execute a function when the form is submitted

Now that we have a working `checkdata()` function, our final step is to call it whenever the form is submitted. We do this using the `onsubmit` event in our HTML code. This event will be triggered when users click the **Submit** button.

The following code links the `onsubmit` event to the `checkdata()` function:

```
<form id="form1" onsubmit="return checkdata()">
```

{:codeblock}

This code ensures that the `checkdata()` function is invoked every time the form is submitted.

Following is the complete code along with the HTML form and JavaScript validation function. Copy and paste the code into your file and check it to determine if it is properly validating:

```
<!DOCTYPE html>
<html>
    <head>
    <title>Contact Details</title>
    <script type="application/javascript">
        function checkdata(){
            //Create references to the input elements we wish to validate
            var username = document.getElementById("name");
            var email_address = document.getElementById("email");
            //Check if username field is empty
            if(username.value == ""){
                    alert("Please enter the name");
                    username.focus();
                    return false;
            }
            //Check if email field is empty
            if(email_address.value == ""){
                    alert("Please enter the email");
                    email_address.focus();
                    return false;
            }
            //If all is well return true.
            return true;
        }
    </script>
    </head>
    <body>
        <h2>Enter your contact details:</h2> <br>
        <form id="form1" onsubmit="return checkdata()">
            <label for="name">Name:</label>
            <input type="text" id="name" name="name">
            <br><br>
            <label for="email">Email Address:</label>
            <input type="text" id="email" name="email">
            <br><br>
            <input type="submit" value="Submit">
            <input type="reset" value="Reset">
        </form>
    </body>
</html>
```

Practice Exercise: Add Javascript interactivity to a HTML page through links

You would have created a web page for Unit Conversion in a file named index.html and styled it with CSS in the lab `Hands-on Lab: Styling your Web Page using CSS` earlier. If not, **click here to complete it**

Now you will add event listeners to the buttons in the web page, to make them functional.

1. Click on the button below to create a new file named script.js.

Open **script.js** in IDE

2. Add the function to convert temparature from Celsius to Fahrenheit

- It should fetch the value inside the “celsius” element in index.html
- The mathematical formula for conversion should be applied

```
Temp(Fahrenheit) = [Temp(Degrees)*9/5] + 32
```

- Set the value property of the “fahrenheit” element in index.html to the returned value

▸ Click here to see the code

3. Add the function to convert weight from Kgs to Pounds

- It should fetch the value inside the “kilo” element in index.html
- The mathematical formula for conversion should be applied

```
Weight(Pounds) = Weight(Kgs) * 2.2
```

- Set the innerHTML property of the “pounds” element in index.html to the returned value

► Click here to see the code

4. Add the function to convert distance from Kms to Miles

- It should fetch the value inside the “km” element in index.html
- The mathematical formula for conversion should be applied

```
Distance(Miles) = Distance(Kms) * 0.62137
```

- Set the innerHTML property of the “miles” element in index.html to the returned value

► Click here to see the code

5. Save the updated script.js.

► Click here to see the completed code in script.js

6. Link style.css to index.html

- Use the `script` tag with the `src` attribute.

► Click to see the code

7. Link the `temperature` section of index.html to the **temperature** function in script.js

- To the button with ID `temperature` in index.html, add an onclick event to invoke the `temperature()` method.
- Assign the ID value of temparature input to be ‘celsius’
- Assign the ID value of temparature output to be ‘fahrenheit’

► Click here to see the code

8. Link the `weight` section of index.html to the **weight** function in script.js

- To the button with ID `weight` in index.html, add an onclick event to invoke the `weight()` method.
- Assign the ID value of weight input to be ‘kilo’
- Assign the ID value of weight output to be ‘pounds’

► Click here to see the code

9. Link the `distance` section of index.html to the **distance** function in script.js

- To the button with ID `distance` in index.html, add an onclick event to invoke the `distance()` method.
- Assign the ID value of distance input to be ‘km’
- Assign the ID value of distance output to be ‘miles’

► Click to see the code

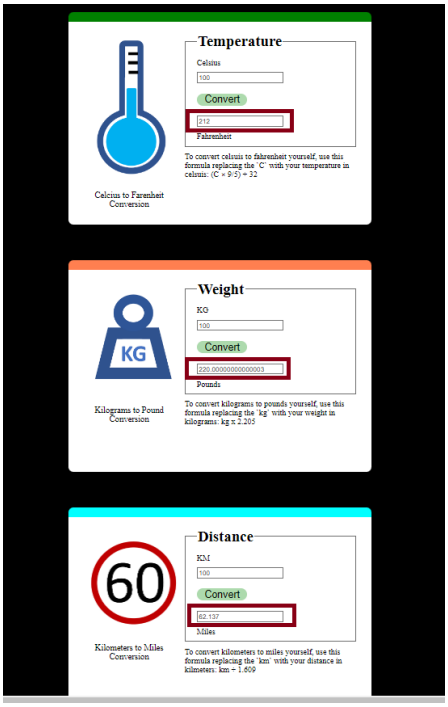
10. Save the code updated in index.html.

▼ Click here to see the compelted code in index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
      <title>Document</title>
      <link rel="stylesheet" href="style.css">
      <script src="script.js"></script>
  </head>
  <body>
      <section id="home">
        <header>
          Unit Conversions
        </header>
        <nav>
          <div class="topdiv">
            <a href="#temperature"><button  class="topmenu">Temperature</button></a>
            <a href="#weight"><button  class="topmenu">Weight</button></a>
            <a href="#distance"><button  class="topmenu">Distance</button></a>
          </div>
        </nav>
      </section>
      <div style="clear:both;"></div>

      <div id="all-conversion-sections">
        <section id="temperature"  class="b temperature">
          <div id="tmp">
            <figure>
              <img src="https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-CD0101EN-SkillsNetwork/labs/Theia%20Labs/02%20-%20HTML5%20Elements/images/thermo.png" width="150px"/>
              <figcaption>Celsius to Farenheit Conversion</figcaption>
            </figure>
            <article>
              <fieldset>
                <legend>Temperature</legend>
                <label for="celsius">Celsius</label> <br/>
<input type="number" id="celsius"> <br/>
                <button id="temperature" onclick="temperature()"> Convert </button> <br/>
                <input type="number" id="fahrenheit"> <br/>
<label for="fahrenheit">Fahrenheit</label>
              </fieldset>
              <aside>
                To convert celsuis to fahrenheit yourself, use this formula replacing the `C` with your temperature in celsuis: (C × 9/5) + 32
              </aside>
            </article>
          </div>
        </section>
        <div style="clear:both;"></div>

        <section id="weight"  class="b weight">
          <div id="wgt">
            <figure>
              <img src="https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-CD0101EN-SkillsNetwork/labs/Theia%20Labs/02%20-%20HTML5%20Elements/images/weight.png"  width="150px"/>
              <figcaption>Kilograms to Pound Conversion</figcaption>
            </figure>
            <article>
              <fieldset>
                <legend>Weight</legend>
                <label for="kilo">KG</label> <br/>
<input type="number" id="kilo"> <br/>
                <button id="weight" onclick="weight()"> Convert </button>  <br/>

                <input type="number" id="pounds">  <br/>
 <label for="pounds">Pounds</label>
              </fieldset>
              <aside>
                To convert kilograms to pounds yourself, use this formula replacing the `kg` with your weight in kilograms: kg x 2.205
              </aside>
            </article>
          </div>
        </section>
        <div style="clear:both;"></div>
        <section id="distance"  class="b distance">
          <div id="dst">
            <figure>
              <img src="https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-CD0101EN-SkillsNetwork/labs/Theia%20Labs/02%20-%20HTML5%20Elements/images/speedo.png"  width="150px"/>
              <figcaption>Kilometers to Miles Conversion</figcaption>
            </figure>
```

```
          <article>
            <fieldset>
              <legend>Distance</legend>
                <label for="km">KM</label> <br/>
<input type="number" id="km">  <br/>
                 <button id="distance" onclick="distance()"> Convert </button> <br/>
                 <input type="number" id="miles">  <br/>
 <label for="miles">Miles</label>
            </fieldset>
            <aside>
              To convert kilometers to miles yourself, use this formula replacing the `km` with your distance in kilometers: km × 0.62137
            </aside>

          </article>
        </div>
      </section>
      <div style="clear:both;"></div>
    </div>
  <div id="go-home" class="iconbutton">
    <a href="#home">
      <img src="https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-CD0101EN-SkillsNetwork/labs/Theia%20Labs/02%20-%20HTML5%20Elements/images/home.svg"/>
    </a>
  </div>
  <footer>Learn more about HTML and CSS as a part of IBM Fullstack Cloud Developer Certification</footer>
 </body>
</html>
```

11. View **index.html** with Live Server and see if you get the converted temperature, weight and distance upon clicking the respective buttons.



Summary

Congratulations! You have now learned how to create a form and validate its user inputs. As a practice exercise, you have added Javascript interactivity to the `Hands-on Lab: Styling your Web Page using CSS` to get the converted values of the `Temparature`, `Weight` & `Distance` metrics.

Tutorial details

Author: Ramesh Sannareddy

Other Contributor(s): Michelle Saltoun