Step 1 of 1

Lab: Validating a JavaScript form

Estimated time 45 minutes

JavaScript is a client-side scripting language and is commonly used to create dynamic web pages. It helps you change web page contents dynamically, as well as enabling 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. Add the `<script>` tag
3. Add a 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

1. Create an HTML form

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

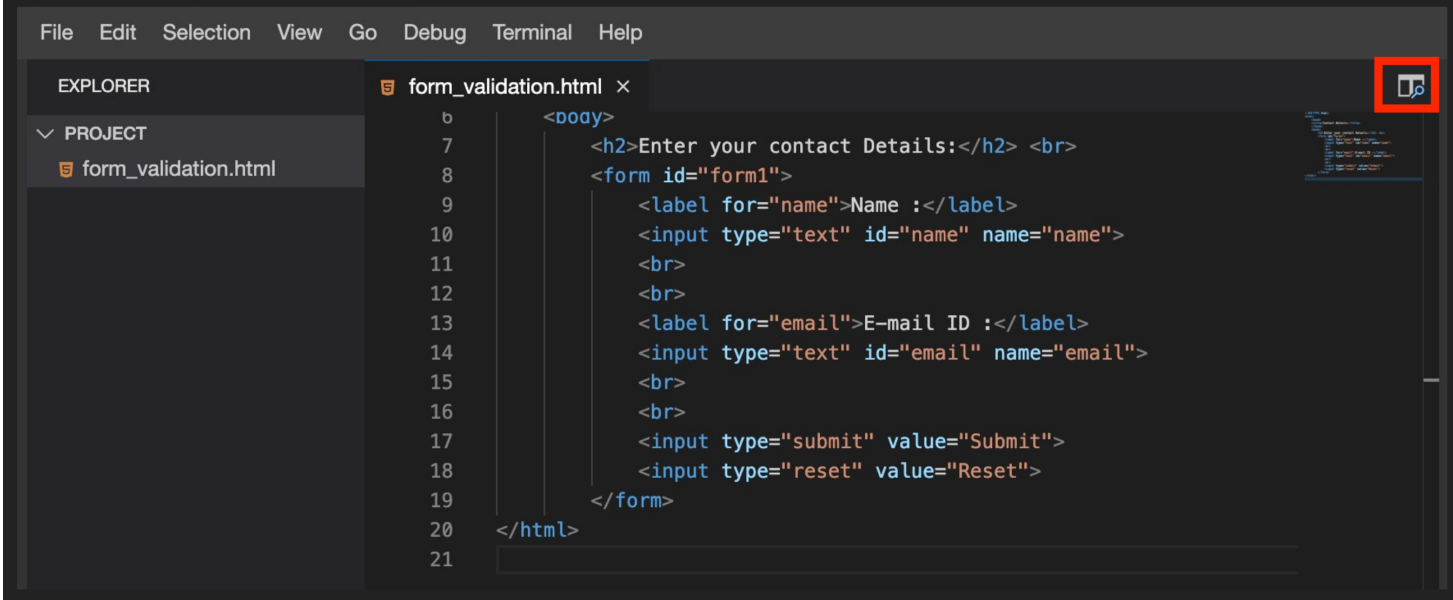
On the window to the right, click on **File > New File**. A **New File** window opens. Enter `form_validation.html` as file name and click **OK**. You are now ready to start creating the new form.

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

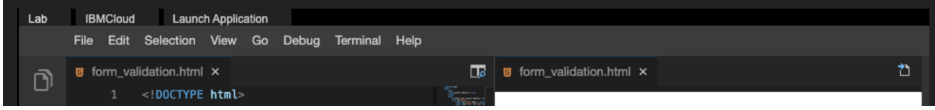
Copy and paste the following code into your file to create the initial form without validation:

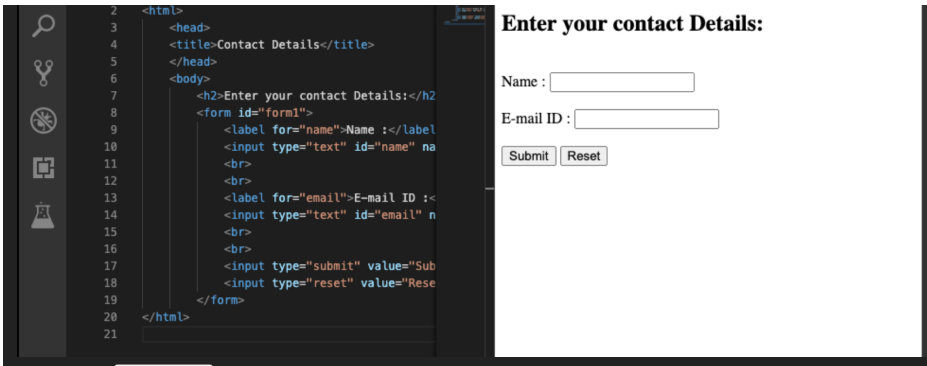
```
<!DOCTYPE html>
<html>
    <head>
    <title>Contact Details</title>
    </head>
    <body>
        <h2>Enter your contact Details:</h2> <br>
        <form id="form1">
            <label for="name">Name :</label>
            <input type="text" id="name" name="name">
            <br>
            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

When you have pasted the code, save your file. To see how your HTML page will display, click the preview icon at the top right of the window.



Your page should look like this:





2. Add 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 to be executed as JavaScript.

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

3. Add a 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`. 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. The references to the elements are stored in two JavaScript variables named `username` and `emailid`.

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

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:

```
        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 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");
            fname.focus();
            return false;
        }
```

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

We 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 JavaScript, we add 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 emailid   = 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(emailid.value == ""){
            alert("Please enter the email");
            emailid.focus();
            return false;
      }
     //If all is well return true.
      return true;
}
```

8. Execute a function when the form is submitted

Our final step is to ensure that the `checkdata` function is executed when the form is submitted. We do this using the `onsubmit` event. This event occurs 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 when 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 emailid   = 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(emailid.value == ""){
                alert("Please enter the email");
                emailid.focus();
                return false;
          }
         //If all is well return true.
         alert("Form validation is successful.")
          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">E-mail ID :</label>
            <input type="email" id="email" name="email">
            <br>
            <br>
            <input type="submit" value="Submit">
            <input type="reset" value="Reset">
        </form>
</html>
```

Summary

Congratulations! You have now learned how to create a form and validate its user inputs. As additional practice, we encourage you to add a numeric field such as "zipcode" and then validate it.

Tutorial details

Author: Ramesh Sannareddy

Change log

Date	Version	Changed by	Change Description
2020-08-13	1.0	Ramesh Sannareddy	Initial version created