Lesson: Form Validation

Form validation

Validating data in the web browser

Since data integrity is essential to most applications, many web forms require specific formats for users to enter data. Ex: A credit card must contain 16 digits, a date cannot have a fifteenth month, and only 50 valid names of states exist for the United States of America.

Data validation is checking input for correctness. While a web server must perform data validation on submitted data, a better user experience occurs when the web browser performs the same data validation before submitting. Any invalid data in the webpage can be immediately flagged as needing modifications without waiting for the server to respond.

Data validation can either be performed while the user enters form data by adding a JavaScript function as the change handler for the appropriate field, or immediately prior to submitting the entire form by adding a function as the form's submit handler.

Example :

click on the following link ,run the code and check:

https://replit.com/join/qhodnqrbyn-pragaticoder (https://replit.com/join/qhodnqrbyn-pragaticoder)

The example mentioned in the link demonstrates :

1. The webpage uses JavaScript in validate.js to validate the web form.
2. The user enters invalid form data and does not check the checkbox.
3. When the user clicks the submit button, the browser executes code in validate.js to validate the form input and highlights invalid fields in red.
4. The user must correct the form data before the browser submits the form data to the web server. After the user clicks the submit button again, the browser updates the page to reflect that all data is valid.

Validating form input with JavaScript

Each textual input element in an HTML document has a value attribute that is associated with the user-entered text. The `value` attribute can be used to validate user-entered text by checking desired properties, such as:

- Checking for a specific length using the `length` property on the `value` attribute
- Checking if entered text is a specific value using `===`
- Checking if the text contains a specific value using the string `indexOf()` method on the `value` attribute
- Checking if the text is a number using `isNaN()`
- Checking that text matches a desired pattern using a regular expression and the string `match()` method

Drop-down menus also have a `value` attribute that is associated with the user-selected menu option.

Checkboxes and radio buttons have a checked attribute that is a boolean value indicating whether the user has chosen a particular checkbox or radio button. The checked attribute can be used to ensure an input element is either checked or unchecked before form submission. Ex: Agreeing to a website's terms of service.

Example :

Consider a web page that asks the user to enter the sales price between $10 to $1000.

The code shall display an error/message for the following condition :

1. The number entered is outside the range

2. The user write the number in text

3. The text field is left blank.

Find the link to the code that satisfies the above stated conditions :

https://replit.com/join/vditctksri-pragaticoder (https://replit.com/join/vditctksri-pragaticoder)

Notice how the if statements are used to write the code

Validating form data upon submission

Validating form data using JavaScript that executes when the user submits the form can be performed by:

1. Register a handler for the form's submit event that executes a validation function.
2. Within the validation function, inspect the form's input fields via the appropriate DOM elements and element attributes.
3. If the form is invalid, call the `preventDefault()` method on the event to cancel the form submission and prevent the form data from being sent to the server.

Example :

Ensuring a checkbox is selected before the form is submitted.

Click the following link to observe and understand the code :

https://replit.com/join/mavntlpjwg-pragaticoder (https://replit.com/join/mavntlpjwg-pragaticoder)

Validating each field as data is entered

Alternatively, form data can be validated as the user enters data in the form by:

1. For each field that should be validated:
    1. Register an input event handler for the field.
    2. Create a global variable to track whether the field is currently valid. In most cases, this global variable should be initialized to false since the form typically starts with the field as invalid.
    3. Modify the global variable as appropriate within the field's event handler.
2. Register a submit event handler for the form that verifies the global variables for each field are true.
3. If one or more of the global variables are false, call the `preventDefault()` method on the submit event to prevent the form from submitting to the server.

The example below uses a regular expression to verify the user enters five digits for the ZIP code. Regular expressions are discussed in more detail elsewhere. The form does not submit unless the ZIP is valid.

Example :

The code below creates a basic web page with a form that allows users to enter a ZIP code. It ensures that the ZIP code is exactly 5 digits long, and if not, it prevents the form from being submitted. This provides a simple client-side validation for the ZIP code input field.

Copy paste the code and check :

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Terms of Service</title>
```

```
  </head>
  <body>
    <form id="tosForm">
      <label for="zip">ZIP:</label>
      <input type="text" id="zip">
      <input type="submit">
    </form>



<script>
let zipCodeValid = false;
let zipCodeWidget = document.querySelector("#zip");
zipCodeWidget.addEventListener("input", checkZipCode);

function checkZipCode() {
  let regex = /^\d\d\d\d\d$/;
  let zip = zipCodeWidget.value.trim();
  zipCodeValid = zip.match(regex);
}

let tosForm = document.querySelector("#tosForm");
tosForm.addEventListener("submit", checkForm);

function checkForm(event) {
  if (!zipCodeValid) {
    event.preventDefault();
  }
}
 </script>
   </body>
   </html>
```

Using HTML form validation

Some HTML form elements and attributes enable the browser to do form validation automatically, which reduces the need for JavaScript validation.

Note
A browser that does not support a particular HTML input element will transform an unsupported

element into a text input, which then requires JavaScript to validate the form data.

Some customized HTML input elements can only contain valid values, such as date or color. Customized elements are automatically checked by the browser and/or filled in by a pop-up input picker in the browser, ensuring the submitted value matches a common specification.

Various element attributes allow the browser to do validation without using JavaScript:

- The required attribute indicates that the field must have a value (text or selection) prior to submitting the form.
- The max and min attributes indicate the maximum and minimum values respectively that can be entered in an input field with ranges, such as a date or number.
- The maxlength and minlength attributes indicate the maximum and minimum length of input allowed by an input field.
- The pattern attribute provides a regular expression that valid input must match.
- The title attribute can be used to provide a description of valid input when using the pattern attribute.

Using HTML form validation.

```
<form>
  <input type="range" name="age" min="5" max="120">
  <input type="checkbox" name="agree" required>
  <input type="password" name="password" minlength="10" maxlength="16">
  <input type="text" name="credit" pattern="^\d{16}$" title="exactly 16 digits">
  <input type="submit">
</form>
```

Several CSS pseudo-classes exist to style input and form elements:

- The :valid pseudo-class is active on an element when the element meets all the stated requirements in field attributes.
- The :invalid pseudo-class is active on an element when one or more of the attributes in the field are not fully met.
- The :required pseudo-class is active on an element if the element has the `required` attribute set.
- The :optional pseudo-class is active on an element if the element does not have the `required` attribute set.