7.7 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.

PARTICIPATION ACTIVITY 7.7.1: Validating form input.

Start 2x speed

```
<!DOCTYPE html>
<html>
<head>
   <title>Purchase Form</title>
   <script src="validate.js" defer></script>
</head>
<body>
   <form>
      <label for="cardNumber">Credit Card #:</label>
      <input type="text" name="cardNumber"
         id="cardNumber">
      <label for="address">Address:</label>
      <textarea name="address" id="address"></textarea>
      <label for="acceptTerms">I accept the terms of
         sale:</label>
      <input type="checkbox" name="acceptTerms"
         id="acceptTerms">
      <input type="submit" value="Submit">
   </form>
</body>
</html>
```

Credit Card #: 12341234
Address:
123 Main St.
Anytown, KS 12345
I accept the terms of sale:
Submit

Captions

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.

Feedback?

PARTICIPATION ACTIVITY 7.7.2: Identify why the field is invalid.

1) Enter 5-digit ZIP code:

ZIP

- Some input characters are not digits.
- The input field is empty.
- The input is too long.

2) Enter 5-digit ZIP code:

103

- Some input characters are not digits.
- The input field is empty.
- The input length is incorrect.

3) Enter 5-digit ZIP code:

31M4N

- Some input characters are not digits.
- The input field is empty.
- The input length is incorrect.

Feedback?

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.

PARTICIPATION ACTIVITY 7.7.3: Validating the sale price.

Start 2x speed

```
<input type="text" id="salePrice">
<span id="errorMsg"></span>
```

Sale price: 9 Must be between $10 and $1000.

```
let salePriceWidget = document.querySelector("#salePrice");
let errorMsg = document.querySelector("#errorMsg");

errorMsg.innerHTML = "";

if (salePriceWidget.value.length === 0) {
   errorMsg.innerHTML = "Sale price is missing.";
}
else if (isNaN(salePriceWidget.value)) {
   errorMsg.innerHTML = "Please enter a number.";
}
else {
   let salePrice = parseFloat(salePriceWidget.value);
   if (salePrice < 10 || salePrice > 1000) {
      errorMsg.innerHTML = "Must be between $10 and $1000.";
   }
}
```

Captions

1. The user can enter a sale price into the input field. The span displays validation error messages.
2. The JavaScript code validates the entered price. If nothing is entered, the code displays an error message next to the text box.
3. If the input is "dog", the next if statement calls isNaN() to detect the invalid input and shows an error message.
4. If the input is a number like 9, parseFloat() converts the "9" string into the number 9.
5. The if statement displays an error message when the price is not between $10 and $1000.

Feedback?

PARTICIPATION ACTIVITY 7.7.4: Using JavaScript to validate input fields.

1) What does the validation function return for `checkGrade("-5")`?

```
function checkGrade(grade)
{
   return grade.length > 0
&& !isNaN(grade);
}
```

- true
- false
- null

2) What does the validation function return for `checkGrade("95.3")`?

```
function checkGrade(grade)
{
   return !isNaN(grade) &&
      grade >= 0 && grade
<= 10;
}
```

- true
- false

3) What does the validation function return for `checkTemperature("-40")`?

```
function
checkTemperature(temp) {
   return temp.length > 0
&& !isNaN(temp) &&
      temp >= 0 && temp <=
1000;
}
```

- true
- false

4) What does the validation function return for `checkTemperature("  ")`?

```
function
checkTemperature(temp) {
   return temp.length > 0
&& !isNaN(temp) &&
      temp >= 0 && temp <=
1000;
}
```

- ○ true
- ○ false

Feedback?

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.

Figure 7.7.1: Ensuring a checkbox is selected before the form is submitted.

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="UTF-8">
      <title>Terms of Service</title>
      <script src="validate.js" defer></script>
   </head>
   <body>
      <form id="tosForm" action="https://example.com" target="_blank"
method="POST">
         <p>
            <label for="tos">I agree to the terms of service:</label>
            <input type="checkbox" id="tos">
         </p>
         <input type="submit">
      </form>
   </body>
</html>
```

```
// validate.js

function checkForm(event) {
   let tosWidget = document.querySelector("#tos");

   // Cancel form submission if tos not checked
   if (!tosWidget.checked) {
      event.preventDefault();
   }
}

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

Feedback?

PARTICIPATION ACTIVITY 7.7.5: Practice validating form prior to submission.

Complete the JavaScript `checkForm()` function so that `checkForm()` sets the input `style.backgroundColor` to `LightGreen` for each field that passes the validation check and sets the input field's `style.backgroundColor` to `Orange` if the validation fails.

Validation rules:

- The screen name field must not be empty.
- The ZIP code field must be of length 5.
- The TOS field must contain "yes".

Note that some browsers will override the light green color with another color if the user chooses an autofill option instead of typing a value.

HTML | CSS | JavaScript

```
<form id="tosForm" action="https://wp.zybooks.com/form-viewer.ph
   <label for="screenName">Screen name:</label>
   <input type="text" id="screenName" name="screenName">
   <label for="zip">ZIP code:</label>
   <input type="text" id="zip" name="zip" placeholder="5-digit Z
   <label for="tos">Type <strong>yes</strong> if you agree to th
   <input type="text" name="agreement" id="tos">
   <input type="submit" value="Submit">
</form>
```

Render webpage Reset code

Your webpage

Screen name:

ZIP code:
5-digit ZIP code

Type **yes** if you agree to the terms of service:

Submit

Expected webpage

Screen name:

ZIP code:
5-digit ZIP code

Type **yes** if you agree to the terms of service:

Submit

▶ View solution

Feedback?

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:
    a. Register an input event handler for the field.
    b. 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.

c. 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.

Figure 7.7.2: Checking a ZIP code field as the user updates the field.

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="UTF-8">
      <title>Terms of Service</title>
      <script src="validate.js" defer></script>
   </head>
   <body>
      <form id="tosForm">
         <label for="zip">ZIP:</label>
         <input type="text" id="zip">
         <input type="submit">
      </form>
   </body>
</html>
```

```
// validate.js

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();
   }
}
```

Feedback?

PARTICIPATION ACTIVITY 7.7.6: Practice validating form as data is entered.

The JavaScript code defines three boolean variables that must all be true for the form to submit: `screenNameValid`, `zipCodeValid`, and `tosAgreeValid`. Two input event handlers exist:

- The `checkScreenName()` function is called when the screen name changes and assigns `screenNameValid` with true when the field is not empty.
- The `checkZipCode()` function is called when the ZIP code changes and assigns `zipCodeValid` with true when the field contains five digits.

Both functions also make the input widget's background orange if the input is invalid.

Add a `checkTosAgree()` function that is called when the TOS agreement input changes. The function should assign `tosAgreeValid` with true when the user enters "yes". The function should also set the input widget's background to the default color if "yes" is typed and orange for all other input.

HTML | CSS | JavaScript

```
1  <form id="tosForm" action="https://wp.zybooks.com/form-viewer.ph
2     <label for="screenName">Screen name:</label>
3     <input type="text" id="screenName" name="screenName">
4     <label for="zip">ZIP code:</label>
5     <input type="text" id="zip" name="zip" placeholder="5-digit Z
6     <label for="tos">Type <strong>yes</strong> if you agree to th
7     <input type="text" name="agreement" id="tos">
8     <input type="submit" id="validate" value="Submit">
9  </form>
10
```

Render webpage Reset code

Your webpage	Expected webpage
Screen name:	Screen name:
ZIP code: 5-digit ZIP code	ZIP code: 5-digit ZIP code
Type **yes** if you agree to the terms of service:	Type **yes** if you agree to the terms of service:
Submit	Submit

▶ View solution

Feedback?

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.

Figure 7.7.3: 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>
```

Feedback?

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.

PARTICIPATION ACTIVITY 7.7.7: Practice with CSS pseudo-classes.

The form below requires all three inputs to be supplied. A red rectangle appears around the form, and the rectangle turns green once all three inputs are given valid values. No JavaScript is used, only CSS pseudo-classes that are automatically applied by the browser.

Make the following modifications:

1. Add the following CSS rule to make each input's background red when the input contains invalid data:

```
input:invalid {
   background-color: #ffdddd;
}
```

2. Add the following CSS rule to make each input's background green when the input contains valid data:

```
input:valid {
   background-color: #ddffdd;
}
```

Render the webpage and verify each input is red until valid data is input, then the input turns green.

HTML CSS

```
1  <form action="https://wp.zybooks.com/form-viewer.php" target="_
2     <label for="dob">Date of birth:</label>
3     <input type="date" name="dob" id="dob" required>
4
5     <label for="creditCard">Credit card number:</label>
6     <input type="text" name="creditCard" id="creditCard" maxleng
7        title="Exactly 16 digits" required>
8
9     <label for="emailAddr">Email address:</label>
10    <input type="email" name="emailAddr" id="emailAddr" required
11
12    <input type="submit">
13 </form>
14
```

Render webpage Reset code

Your webpage

Date of birth:
yyyy/mm/dd

Credit card number:

Email address:

Submit

Expected webpage

Date of birth:
yyyy/mm/dd

Credit card number:

Email address:

Submit

▶ View solution

Feedback?

PARTICIPATION ACTIVITY 7.7.8: Form validation questions.

1) If all the fields in a form have been validated before submitting the form data to a server, does the server need to repeat the field validation?

- Yes
- No

2) Is validating input fields as the user fills in each field better than validating the entire form after all the form data has been entered?

- Yes
- No

3) If validation shows that a form input value is invalid, should the input value be reset to the initial value?

- Yes
- No

4) Can web developers do all form data validation using HTML input element attributes and not use JavaScript validation?

- Yes
- No

Feedback?

CHALLENGE ACTIVITY 7.7.1: Form validation.

530096.4000608.qx3zqy7

Jump to level 1

Use HTML validation attributes to ensure the entered age is between 21 and 65, inclusive, and the username is 15 characters or less. **SHOW EXPECTED**

```
1 <form>
2    <p>
3       <label for="userAge">User Age:</label>
4       <input id="userAge" type="number" name="age" min="21" max=
5    </p>
6    <p>
```

1 2 3 4

```
7     <label for="userName">Username:</label>
8     <input id="userName" type="text" name="username" maxlength
9   </p>
10  <input type="submit">
11 </form>
```

5

1 | 2 | 3 | 4 | **5**

Check | Next

Done. Click any level to practice more. Completion is preserved.

✓

✓ Testing min attribute of userAge input field

Yours `21`

✓ Testing max attribute of userAge input field

Yours `65`

✓ Testing maxlength attribute of userName input field

Yours `15`

Your webpage

User Age:

Username:

Submit

Feedback?

Exploring further:

- Form data validation from MDN

How was this

Provide section feedback

section?