INT222 - Internet Fundamentals

Week 9: Using JS in HTML, Client-side validation

Agenda

- Using JavaScript in HTML Page
- Introduction to Client-side validation
  - Using HTML5 Features
  - Using JavaScript

Using JavaScript in HTML Page

- The ways of adding JavaScript code to HTML page:
  1. Inline (embedded) JavaScript code: Basic event handlers.
  2. Internal JavaScript code: Using script tags.
  3. External JavaScript code: Using code stored in a separate .js file.

Inline (embedded) JavaScript code

- Event Handlers: Whenever an event happens on a page, the browser detects it. Scripts that handle events are referred to, appropriately, as event handlers.
- e.g.

  <input type="button" id="hello" value="Hello" onClick = "window.alert('Hello World!')“>

  Notes: <input> without a <form> appears valid

  - Js-code-inline.html

Internal JavaScript code

- Internal JavaScript code: Using <script> tags.
  - type attribute is optional because "text/javascript" is its default value.
- Scripts can be inserted anywhere on a page:
  - in <head></head> e.g.

```
<script>
    function sayHello()    {  alert("Hello!"); }
</script>
```

  - in <body></body> e.g.

```
<script>
    today = new Date();
    document.write(today); // write into page in current location
</script>
```

- js-code-internal.html

External JavaScript code

- Using code stored in a separate **.js** file
- e.g
  - js-code-external.html
  - external.js
- Note: jQuery is not covered in INT222.

Where to put JavaScript?

- JavaScript code can be added anywhere within <head></head> or <body></body>.

- For large JavaScript libraries, JavaScript code should be placed just before the ending tag of body element.
  - Why?

Play around with JavaScript & HTML

- Using Element innerHTML Property
  - innerHTML.html
- Changing CSS with JavaScript
  - changeCSS.html
- Temperature Convertor
  - temp-conversion.html

Introduction to Client-side Form Validation

Client-Side Form Validation

- At the client-side of an web app, validate and ensure the user's form inputs are necessary and properly formatted for form processing.
- Advantages
  - Saves time and bandwidth.
  - It's fast with immediate user feedback without having to wait for the page to load.
  - You can safely display only one error at a time and focus on the wrong field, to help ensure that the user correctly fills in all the details as required.
  - Still need server-side validation.
  - Client and server-side validation complement each other, and as such, they really shouldn't be used independently.

Client-Side Validation with HTML5

- HTML5 provides several new types for form <input> tags.
  - These new features allow better input control and validation.
  - Some HTML5 new values of input type attribute:

    color, date, datetime, email, month, number, range, search, tel, time, url, week

- input-tags-html5.html

Client-Side Validation with HTML5

- required attribute
  - Specifies that an input field is required (must be filled out).
  - but spaces are acceptable.
  - For radio buttons, checkboxs and select-option, The required attribute is not supported in any of the major browsers.
- pattern attribute
  - Specifies a regular expression to check the input value against.
    - E.g. Phone Number (format: xxx-xxx-xxxx):

```
<label for="phone">Phone Number:</label>
<input type="tel" pattern="^\d{3}-\d{3}-\d{4}$" id="phone">
```

    - Attribute pattern is only allowed when the input type is email, password, search, tel, text, or url.

Client-Side Validation with HTML5

- min, max, maxlength, step attributes
  - Specifies the minimum/maximum value for number, date or range input field
  - e.g.

```
<input type="number" name="entry12"
        min="2" max="20" step="2" />
```

Client-Side Validation with HTML5

- title attribute
  - Used to give hints, show validation rules or instructions
  - Show up when move and shop the cursor on the elements.
  - e.g.

```
SSN: <input type="text" name="ssn"
            pattern="^\d{3}-\d{2}-\d{4}$"
            title="The Social Security Number" />
```

- validation-html5.html

Client-side Validation Using JS

- Approaches
  1. Display the errors one by one, focusing on the offending field.
     - Makes revising and successfully submitting the form much easier for the user
  2. Display all errors simultaneously, server-side validation style.

Client-side Validation Using JS

- Guidelines
  - Presence or Absence Test
    - To determine whether the required fields left empty.
  - Value Test
    - To determine if a field has a specific value or code.
  - Range Test
    - To determine if a value entered is within a specific range (inclusive or exclusive)

Client-side Validation Using JS

- Guidelines (cont')
  - Reasonableness Test
    - To determine if a value entered is reasonable based on other information supplied or information available to us. This test needs to be review periodically.
  - Check Digit Test
    - To determine if for example, a credit card number or a Driver's license number is valid.
  - Consistency Test MULTIPLE FIELD(s)
    - To determine if a value entered is consistent with other information entered.

JavaScript Validation

- HTML form onsubmit event attribute
  - Execute a JavaScript when a form is submitted.
  - The browser will stop sending the form to server only when the onsubmit attribute (event handler) gets the value of "return false".
- e.g.

```
<form id='example' name='example' method='post'
        action='https://somesite/cgi-bin/echo-p.pl'
        onsubmit='return formValidation();'>
    …. ….
</form>
```

- Note: never use onsubmit on the submit button. That will not stop the invalid data to be send out.

Example - Validating Text Field

- Rule: all digits
- Code:

```
function validatePhoneNumber() {
    var input = document.form1. phone.value.trim();
    if (parseInt(input) != input) {
        alert('Please enter a phone number, numbers only');
        document.form1. phone.focus();
      return false; // failed for validation
    }
    return true; // passed for validation
 }  //  End of function
```

- Note: don't use RegExp in JS validation for this course.

❑ js-form-validation-all-digits.html

Example - Validating Text Field

- Rule: all alphabetic letters (‘a’-’z’, ‘A’-’Z’)

```
function validateSurname() {
    var allAlpha = true;
    var elem = document.getElementById("client");
    var inputValue = elem.value.trim();
        inputValue = inputValue.toUpperCase();
    for (var i = 0; i < inputValue.length; i++) {
        // check all character are letters
        if (inputValue.charAt(i) < "A" || inputValue.charAt(i) > "Z" )  { allAlpha = false; }
    }  // for

    if (!allAlpha){
        alert("Name : Please enter a meaningful name with all alphabet letters.");
        elem.focus();
        return false;
    } /* else */
    return true;
}  // function
```

- js-form-validation-all-alphabetic-letters.html

Example - Validating Text Field

- Rule: (contains) at least one alphabetic letter ('a'-'z', 'A'-'Z')

```
function validateSurname(frm) { // pass in form object in HTML
   var passAlpha = false;
   var alphString = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
   var inputValue = frm.surname.value.trim();
   var inputValue = elem.value.trim();
   for (var i = 0; i < inputValue.length; i++) {
         // check at least one character is a letter
         if (alphString.indexOf(inputValue.substr(i,1))>= 0)   { passAlpha = true; }
      }  // for
   if (!passAlpha){
         alert("Name : Please enter a meaningful name with at least one Alphabet letter.");
         frm.surname.focus();
         return false;
      } else { return true; }
}  // function
```

- js-form-validation-at-least-1-letter.html

Examples - Validating Multiple Fields/Rules

- Example 1
  - Show error messages using alert().
  - js-form-validation-name-and-phone.html
- Summary: text field objects can be assessed
  - using getElementById
    - document.getElementById("elementid").value
  - using form name and form control/element name
    - document.formname.elementname.value
  - using this keyword with passing form object in HTML
    - passedInForm.elementname.value
  - more …

Examples - Validating Multiple Fields/Rules

- Example 2
  - Show error messages on the web page:
  - ❑ js-form-validation-multipleField-error-on-page.html
- Validation rules used:
  - Validating name:
    - must present; minimum 4; all alphabetic letters
  - Validating phone number:
    - must present; in the format: ###-###-####
  - Error message: showed on web page.
  - Code: next page
- Notes: - no "else-if" is used → for easy coding but only one error message is showed at a time for each field.

Validating textarea

- Rule: presence, not only whitespace(s)

```
function validateTextarea(form1)   {
  /* Validate that the textarea named "comments" in the form named
       "form1" has some text. */
  if (form1.comments.value.trim().length == 0) {// check length of textarea
        alert("No input! Please enter your comments.\n");
        form1.comments.value = "";
        form1.comments.focus();
        return false;
    }
    return true;
} // function
```

- form-validation-textarea.html

Validating – radio button

- Rule:
  - must select one
- To determine which one is checked:

  if (document.formname.radioname[i].checked)
- e.g.

```
var checked = false;
for (var i = 0; i < radio_num; i++)  {
    //if (document.formname.radioname[i].checked== true)
    if (document.formname.radioname[i].checked)  {
        checked = true;
    }
 }
```

- form-radio-validation.html

Validating checkbox

- Rules:
  - At least check one
  - Check all of the boxes
  - Check none of the boxes
- To determine which one is checked:

  if (document.formname.checkboxname[i].checked)
- e.g.

```
for (var i = 0; i < radio_num; i++)  {
  //if (document.formname.checkboxname[i].checked== true)
  if (document.formname.checkboxname[i].checked)  {
      counter++;
  }
}
```

- form-validation-checkbox.html

Validating select/option: Single Selection

- Select options logic
  - Get the selectedIndex:

```
var x = document.example.whatToDo.selectedIndex;
```

  - If selectedIndex == -1
    - None are selected
  - If the selectedIndex is NOT -1
    - document.example.whatToDo.options[x].value;
      - for the value
    - document.example.whatToDo.options[x].text;
      - for the text
- form-validation-select-single.html

Text vs Value

- In select-option controls, we may have both text and value. It's the value will be sent to the server.

```
<select>
  <option value="This is a value">This is the text</option>
  <option value="This is a value " selected>
      This is text
  </option>
</select>
```

- If value attribute is not provided, the text is the value.

Validating select/option: Multiple Selection

- Get the number of the select options using length

  document.form1.selname.options.length;

- Loop to check which one was selected

  if (document.form1. selname[i].selected == true) //selected

- Read option value and text

  document.form1. selname[i].value

  document.form1. selname[i].text

- form-validation-select-multiple.html

Validation using JavaScript Summary

- onsubmit:

```
<form method='post' name='form1'
        action = "http://formpost.azurewebsites.net/home/test"
        onsubmit='return validateFrom()'>
```

- Refer to the element:

```
document.formname.elementname
  e.g. document.form1.name.value.trim()
if (document.form1.specialty[i].checked) {…}
if (document.form1.plans.selectedIndex == -1) {…}
```

Validation using JavaScript Summary

- Refer to the element (cont'd):
  - using getElementById
    - document.getElementById("elementid").value
  - using form name and form control/element name
    - document.formname.elementname.value
  - using this keyword with passing form object in HTML
    - passedInForm.elementname.value
  - more …
- Validation function returns
  - True/false
  - Notes: only "return false" can stop sending the form to server. So if you validation code has syntax error(s), the form will always be sent out.

Thank You!