COURSE TECHNOLOGY
CENGAGE Learning

JavaScript, Sixth Edition

Chapter 6
Enhancing and Validating Forms

Objectives

When you complete this chapter, you will be able to:

- Enhance form usability with JavaScript
- Customize browser-based HTML validation
- Implement custom validation to check for errors and display error messages

Using JavaScript with Forms

- Validation
  - checking that form information provided by users conforms to data rules
- `form` object
  - Represents a form in an HTML document
  - Used to access form and its data

Using JavaScript with Forms (cont'd.)

PROPERTY	DESCRIPTION
`autocomplete`	Enables autocompletion of previously saved form data by a browser when set to `true`
`elements`	Returns a collection of a form's elements
`length`	Returns an integer representing the number of elements in the form
`novalidate`	Disables browser-based validation by a browser when set to `false`

Table 6-1 Properties of `form` objects

EVENT	DESCRIPTION
`submit`	Fires when a form's submit button is clicked

Table 6-2 Event of `form` objects

METHOD	DESCRIPTION
`checkValidity()`	Initiates browser-based validation of form controls, returning `true` if all controls are valid
`submit()`	Submits a form without the use of a submit button

Table 6-3 Methods of `form` objects

Using JavaScript with Forms (cont'd.)

- Common elements for collecting form data:
  - `input`
  - `select`
  - `option`
  - `textarea`
  - `button`

Working with Input Fields (cont'd.)

PROPERTY	DESCRIPTION
`autofocus`	Returns the value `true` if the HTML autofocus attribute is set, indicating that the element should receive the focus when the form is loaded
`placeholder`	Returns the value of the `placeholder` attribute, which contains text to be displayed when a field has no value
`required`	Returns the value `true` if the HTML `required` attribute is set, indicating that the control must contain a value before the form can be submitted
`validationMessage`	Sets or returns the text of the message to be displayed to the user after a failed `submit` event if the field's `validity` value is `false`
`validity`	Returns the value `true` if the control value passes browser-based validation rules
`value`	Sets or returns the value of the control
`willValidate`	Returns the value `true` if constraint validation is enabled for the control

Table 6-4 Properties of elements within forms

Working with Input Fields (cont'd.)

METHOD	DESCRIPTION
`checkValidity()`	Runs constraint validation against a form element; returns a value of `true` if the value is valid, and `false` if the value is invalid
`setCustomValidity("message")`	Sets the string *`message`* as the text to be displayed to users if the control value is found to be invalid; passing a nonempty string sets the control's validity to `false`, and passing an empty string as the parameter sets the control's validity to `true`

Table 6-5 Methods of elements within forms

EVENT	TRIGGERED WHEN
`blur`	The focus leaves the element, meaning that the element is initially selected or contains the insertion point, and then another element is selected or contains the insertion point (usually by clicking another element or pressing Tab to move to another element)
`change`	The focus leaves the element, and the value or selected state of the current element has changed
`focus`	The element receives the focus, meaning the element is selected or the insertion point moves into the current element (usually by clicking the current element or pressing Tab to move to the current element)
`formchange`	Another control in the form fires a `change` event
`forminput`	Another control in the form fires an `input` event
`input`	The value or selected state of the current element changes
`invalid`	A control's value is found to be invalid during constraint validation

Table 6-6 Events of elements within forms

Referencing Forms and Form Elements

- Can use `getElementsByTagName()` method:

```
getElementsByTagName("form")[0]
```

- `Document` object includes a `forms[]` array
  - Contains all forms on a web page
- `form` object has an `elements[]` array

Referencing Forms and Form Elements (cont'd.)

- `elements[]` array
  - Contains objects representing each control in a form
- Reference form index number in the `forms[]` array
  - Followed by the appropriate element index number from the `elements[]` array

Improving Form Usability

- Before validation
  - Can reduce amount of validation necessary

Designing Forms to Collect More Accurate Content

- Replace input boxes with other fields that present limited choices

Designing Forms to Collect More Accurate Content (cont'd.)

NAME	CODE TO CREATE	USE TO DISPLAY
Option buttons	`<input type="radio" />`	A small set of options at once, from which a user can select one
Check boxes	`<input type="checkbox" />`	One or more yes/no choices

Selection lists	`<select>`
	`<option>value1</option>`
	`<option>value2</option>`
	`...`
	`</select>`
		A truncated list of options that's fully displayed when a user interacts with it

Sliders	`<input type="range" />`	A bar with an indicator that users can drag to increase or decrease a value; used with `min` and `max` attributes to specify bottom and top of range (supported in IE10+ and all other modern browsers)

Data lists	`<input type="text" list="listname" />`
	`<datalist id="listname">`
	`<option value="value1" />`
	`...`
	`</datalist>`
		A text box that suggests values from the `datalist` element as user types, but enables a user to enter a value not in the list

Table 6-7 Selected form elements for providing limited choices

Designing Forms to Collect More Accurate Content (cont'd.)

Figure 6-2 Sample fieldset containing `input` elements

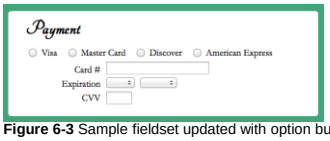

Figure 6-3 Sample fieldset updated with option buttons and selection lists

Programming Forms to Increase Content Accuracy

- Assistive functions
  - Reduce likelihood of user errors
  - Prevent users from entering erroneous data
- Removing default values from selection lists
  - Can set default value for selection list in HTML
    - Only to one of the options
  - JavaScript can set `selectedIndex` property to -1
    - Corresponds to no selection

Programming Forms to Increase Content Accuracy (cont'd.)

PROPERTY	DESCRIPTION
`length`	Returns the number of `option` elements nested within the `select` element
`multiple`	Sets or returns a Boolean value that determines whether multiple options can be selected in the selection list
`options`	Returns a collection of the elements nested within the `select` element
`selectedIndex`	Returns a number representing the element number in the `options` collection of the first option selected in a selection list; returns –1 if no option is selected
`size`	Sets or returns the number of options to be displayed at once
`type`	Returns the type of selection list, which is either `select-one` if the `select` element does not include the `multiple` attribute, or `select-multiple` if the `select` element does include the `multiple` attribute

Table 6-8 `select` element properties

Programming Forms to Increase Content Accuracy (cont'd.)

- Dynamically Updating Selection List Values
  - Can add or remove `option` elements from a `select` element using node methods
    - Can change list options based on selection in another field

Programming Forms to Increase Content Accuracy (cont'd.)

PROPERTY	DESCRIPTION
`defaultSelected`	Returns a Boolean value that determines whether the `option` element representing the currently selected item includes the `selected` attribute
`index`	Returns a number representing the element number within the `options` collection
`label`	Sets or returns alternate text to be displayed for the option in the selection list
`selected`	Sets or returns a Boolean value that determines whether an option is selected
`text`	Sets or returns the text displayed for the option in the selection list
`value`	Sets or returns the text that is assigned to the `option` element's `value` attribute; this value is submitted to the server

Table 6-9 Properties of `option` elements

Programming Forms to Increase Content Accuracy (cont'd.)

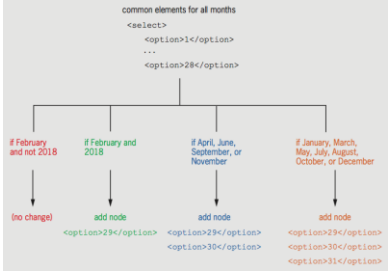

Figure 6-5 Diagram of function for dynamically updating selection list values

Programming Forms to Increase Content Accuracy (cont'd.)

- Adding Placeholder Text for Older Browsers
  - `placeholder` attribute of `input` and `textarea` elements
    - Supported by modern browsers
    - Can recreate behavior with JavaScript for older browsers:
      - Add placeholder text when page finishes loading
      - Remove placeholder text when user selects field
      - Add back placeholder text if user makes no entry

Programming Forms to Increase Content Accuracy (cont'd.)

- Automatically updating an associated field based on a user entry
  - Multiple elements may be associated
    - Example: check box to indicate `textarea` entry
  - Can automatically change value of one field in response to change in other field

Programming Forms to Increase Content Accuracy (cont'd.)

- Transferring duplicate field values
  - Can copy data from one field to another based on user indicating they should have the same value
    - Example: Shipping Address and Billing Address

Programming Forms to Increase Content Accuracy (cont'd.)

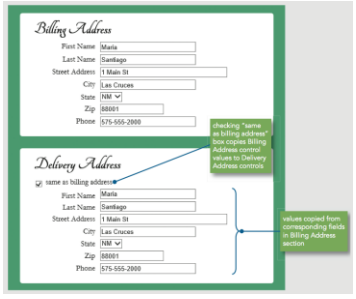

Figure 6-10 Billing Address entries copied to Delivery Address section

Customizing Browser-Based Validation

- Modern browsers can perform some validation
  - Known as browser-based validation, native validation, or HTML5 validation

Customizing Browser-Based Validation (cont'd.)

- Specifying browser-based validation parameters
  - Use attributes listed in Table 6-12

ATTRIBUTE	DESCRIPTION	USE WITH
formnovalidate	Toggles off validation of the form when added to the <input> tag for a submit button	input elements with a type value of submit
max	Specifies the control's maximum numerical value	input elements with a type value of number
maxlength	Specifies the control's maximum number of characters	textarea elements, or input elements displayed as a text box
min	Specifies the control's minimum numerical value	input elements with a type value of number
novalidate	Toggles off validation of the form when added to the opening <form> tag	the form element
pattern	Specifies a pattern that a control's value must match, expressed as a regular expression	textarea elements, or input elements displayed as a text box
required	Indicates that the control must have a value	input, select, or textarea elements
step	Specifies the increment that a numerical value must adhere to	input elements with a type value of number

Table 6-12 HTML attributes to set browser-based validation parameters

Customizing Browser-Based Validation (cont'd.)

- Specifying browser-based validation parameters
  - Additional validation linked to `input type` values

VALUE	CONTENT DESCRIPTION	SAMPLE CONTENT
color	# followed by a 6-digit hexadecimal color value, with any letters in lower case	#ffcc00
date, datetime, week, month, time, datetime-local	Relevant components of coordinated universal time (UTC), a standardized date/time format	2019-10-14T12:00:00.001-04:00
email	An email address	president@whitehouse.gov
number	A numerical value	73.54

Table 6-13 Values for `type` attribute that trigger browser-based validation

Customizing Browser-Based Validation (cont'd.)

- Customizing browser-based validation feedback
  - Modern browsers display feedback in similar ways, with variation
    - Displayed after `submit` event triggered
    - Invalid controls highlighted
    - Bubble displayed next to first control

Customizing Browser-Based Validation (cont'd.)

- Customizing browser-based validation feedback (cont'd.)
  - Customizable through constraint validation API
    - All properties of `validity` object must have value of false for element to be valid

Customizing Browser-Based Validation (cont'd.)

PROPERTY	RETURNS `true` IF
customError	A custom error message has been set with `setCustomValidity()`
patternMismatch	The control value does not match the value of the `pattern` attribute
rangeOverflow	The control value is greater than the value of the `max` attribute
rangeUnderflow	The control value is less than the value of the `min` attribute
stepMismatch	The control value does not conform to the value of the `step` attribute
tooLong	The length of the control value is greater than the value of the `maxlength` attribute
typeMismatch	The control value does not conform to the rules for the `type` attribute value
valueMissing	The control value is empty but the control has a `required` attribute set
valid	None of the preceding properties are `true`

Table 6-14 `validity` properties

Customizing Browser-Based Validation (cont'd.)

- Customizing browser-based validation feedback (cont'd.)
  - `checkValidity()` and `setCustomValidity()` methods
  - CSS `:invalid` and `:valid` pseudo-classes
    - Use to change properties of form elements based on validity status

Customizing Browser-Based Validation (cont'd.)

- Customizing browser-based validation feedback (cont'd.)

JavaScript

```
var fname = document.getElementbyId("firstName");
if (fname.valueMissing) {
  setCustomValidity("Please fill out this field.");
}
```

CSS

```
#firstName:invalid {
   background: rgb(255,233,233);
}
```

Customizing Browser-Based Validation (cont'd.)

- Customizing browser-based validation feedback (cont'd.)

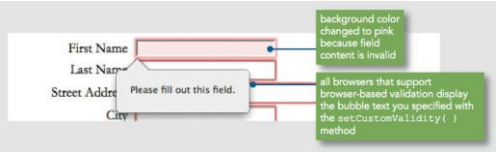

Figure 6-13 Customized browser-based validation

Customizing Browser-Based Validation (cont'd.)

- Customizing browser-based validation feedback (cont'd.)
  - Bubble appearance varies among browsers
  - Cannot set multiple validation messages for a single field at once
  - Can disable browser-based validation using the `preventDefault()` method and the `invalid` event
    - If disabled, must program custom validation

Programming Custom Validation

- Common validation functions:
  - Checking that required fields contain entries
  - Checking values dependent on other fields
  - Checking for appropriate content type

Validating Submitted Data

- `submit` event fires when a form is submitted
  - Often when submit button selected
  - Data usually validated when `submit` event fires
  - `preventDefault()` method disables default behavior of an event when it fires
    - Not supported in IE8, so set `returnValue` to `false` instead

Validating Required Fields with Custom Functions

- Retrieve values of required fields, then check if any is empty

```
try {
   if (element.value === "") {
      throw "message";
   }
}
catch(message) {
   // code to display message and highlight error
   formValidity = false;
}
```


Validating Required Fields with Custom Functions (cont'd.)

- Checking for empty text input fields
  - Check `value` property for a value

```
if (document.getElementById("firstName").value === "") {
  // code to run if the field is blank
}
```

  - Use loop statement to check each field in a group

Validating Required Fields with Custom Functions (cont'd.)

- Checking for selection lists with no values
  - Check value of `selectedIndex` property
    - If no option is selected, value is -1

```
if (document.getElementById("state").selectedIndex === -1 {
    // code to run if the field is blank
}
```


Validating Required Fields with Custom Functions (cont'd.)

- Checking for option button sets with no selection
  - Check value of `checked` property
  - Use And (`&&`) operators to check if no option button is selected

```
var buttons = document.getElementsByName("Color");
if (!buttons[0].checked && !buttons[1].checked && ↵
    !buttons[2].checked) {
    // code to run if no button is selected
}
```


Validating Dependent Fields with Custom Functions

- Sometimes need to test logic specific to a form
- Validating based on the state of a check box
  - Access same `checked` property used with option button
- Validating based on text input box contents
  - Can use nested `if` statements to account for possibilities when entry in one text box requires entry in another

Validating Dependent Fields with Custom Functions (cont'd.)

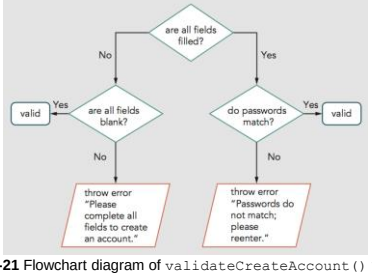

Figure 6-21 Flowchart diagram of `validateCreateAccount()` function

Validating Content Type with Custom Functions

- Can check whether numeric fields contain numbers
  - Use `isNaN()` function
    - returns true if value is not a number

```
isNaN(document.getElementById("subtotal").value)
```

- Character patterns like zip codes require regular expressions (Chapter 8)

Summary

- Validation checks that information conforms to rules
- Assistive functions reduce likelihood of user errors
- Browser-based validation is built into modern browsers
  - Customizable through Constraint Validation API
- `preventDefault()` method blocks action normally associated with an event

Summary (cont'd.)

- To validate required text input fields
  - Retrieve the values of the required fields
  - Check if the value of any of them is an empty string
- To validate required selection lists
  - Retrieve the `selectedIndex` value
  - Check whether it's equal to -1

Summary (cont'd.)

- To check if an option button is selected, access the value of its `checked` property.
- To check if none of the option buttons in a set are selected, create a conditional statement using And (`&&`) operators
- In some cases, you need to create validation functions to test logic specific to your form