Scripting Forms with JavaScript

CSE/ISE 102: Intro to Web Design

Stony Brook University

Form Representation

- In JavaScript, represents forms with the HTMLFormElement type
- In addition to the normal HTMLElement properties, we also have:
  - action — URL to send form request to
  - elements — collection of all controls in the form
  - length — # of controls in the form
  - method — type of HTTP request to send (get or post)
  - submit() — submits the form

Retrieving Form Elements

- Use `getElementById()` like we do with other page elements
- We can also use `document.forms`, which is a collection of all the forms on the page
  - Use a numerical index or the form *name* (not the ID)
    - e.g., `document.forms[0]`
    - e.g., `document.forms["myForm"]`

Form Submission

- The submit event fires right before the request is sent to the server
  - we can intercept this event to validate the form and decide whether to allow the form submission
  - cancel the form submission by calling `preventDefault()` on the event (which is passed in as the sole argument of the function
- btn.onsubmit = function(event) { event.preventDefault(); ... };

Form Fields

- Access these as normal using DOM methods
- Each form also has a collection named elements that contains these controls
  - form elements appear in order
- If multiple form controls use the same name (e.g., radio buttons), then the corresponding element is a collection of those elements

```
<ul>
    <li>
        <input type="radio" name="color" value="red">
        Red
    </li>
    <li>
        <input type="radio" name="color" value="green">
        Green
    </li>
    <li>
        <input type="radio" name="color" value="blue">
        Blue
    </li>
</ul>

var colorFields = form.elements["color"];
var firstColor = colorFields[0];
```

Common Form Field Properties

- disabled — Boolean value
- name — name of the field
- readOnly — a Boolean value
- type — type of the field ("checkbox", "radio", etc.)
- value — the value that will be submitted to the server
- The fields above can be changed dynamically

Preventing Multiple Submissions

- Use the `disabled` property to prevent the user from submitting the same form multiple times
  - e.g., in case the server is taking too long to respond
- Get the button from the form element, disable it, then call submit() on the form

Two More Common Methods

- focus() — sets browser's focus to the form field
  - this is often done when a page is loaded
  - onload="document.forms[0].elements[0].focus();"
- blur() — removes focus from an element
  - used to be used to skip read-only fields
  - not really used anymore

Common Form Field Events

- blur — fires when field loses focus
- change — fires when field loses focus and value has changed
  - used for input, textarea, and select elements
- focus — fires when field gets focus

Scripting Text Boxes

- select() — selects all of the text in a text box
  - most browsers automatically focus on that text box
  - onfocus = function(event){ event.target.select(); };
- `select` event fires when text box is selected
  - use `value` field to get actual text
  - HTML5 adds selectionStart and SelectionEnd properties

Input Filtering

- We want text boxes to be able to require the presence of (or the absence of) specific characters
- Use the keypress event to examine character codes and decide whether or not to allow them

Blocking All Input

- onkeypress = function(event) { event.preventDefault(); };

Blocking Non-Digits

- var charCode = event.getCharCode()
  if (!/\d/.text(String.fromCharCode(charCode)) && charCode > 9)
  {
  event.preventDefault();
  }

Regular Expression Summary

- Surrounded by / and /
- . = any character
- * + ? mean 0+, 1+, or 0 or 1
- character matches that literal character
- [ ] surrounds options (e.g., [bc] means "b or c")
- Use backslash to escape metacharacters: ([{ \ ^ $ |)] } ? * + .

Text Boxes: Tabbing Forward

Scripting Select Boxes

- Adding elements
  - use DOM methods to create new "option" elements
  - use select box's add() method:
    - var newOption = new Option("Option text", "Option value");
      selectbox.add(newOption, undefined); // add at end of list

Removing Select Box Options

- Use the remove() function to remove a specified index
- for (var i = 0, len = box.options.length; i < len; i = i)
  {
  box.remove(0);
  }