Forms, Form Events, and Validation

Bok, Jong Soon
jongsoon.bok@gmail.com
https://github.com/swacademy

How to Access to Form

- In JavaScript, access forms through the Document Object Model (DOM).
- The first approach is to use the forms property on the document object.

```
var theForm = document.forms[0];
```

- The problem is that if you change the page—if you add or remove a form—your JavaScript no longer works.

```
<form id="someform" ...>
var theForm =
document.getElementById("someform");
```

Attaching Events to Forms: Different Approaches

- The primary event is **`submit`**, and the event handler is **`onsubmit`**.
- To attach the event handler to the form using the traditional method:

```
document.getElementById("someform")
                        .onsubmit = formHandler;
```

- To attach an event handler inline to the form, incorporate it into a **`return`** statement:

```
<form name="someForm"
        onsubmit="return formHandler();">
```

Cross-Browser Event Handling

- When want to use **`attachEvent`** and **`addEventListener`** for cross-browser compatibility.
- A better approach is to create a reusable event handling function, such as the following:

```
<script type="text/javascript">

 function catchEvent(eventObj, event, eventHandler) {
    if (eventObj.addEventListener) {
       eventObj.addEventListener(event, eventHandler,false);
    } else if (eventObj.attachEvent) {
       event = "on" + event;
       eventObj.attachEvent(event, eventHandler);
    }
 }

</script>
```

Canceling an Event

- To stop the form submission
  - `cancelBubble` to true (for MSIE) and the `returnValue` property to `false`.
  - `preventDefault()` method call (other browsers) in combination with `stopPropagation()`.

```
<script type="text/javascript">

function formFunction(evnt) {
    var event = evnt ? evnt : window.event;

    if (event.preventDefault) {
        event.preventDefault();
        event.stopPropagation();
    } else {
        event.returnValue = false;
        event.cancelBubble = true;
    }

</script>
```

Canceling an Event (Cont.)

- The `returnValue` property is equivalent to returning `false` explicitly in the function.
- The `preventDefault` prevents the default behavior based on the object and the event.

```
<script type="text/javascript">

function cancelEvent(event) {
    if (event.preventDefault) {
        event.preventDefault();
        event.stopPropagation();
    } else {
        event.returnValue = false;
        event.cancelBubble = true;
    }
}

</script>
```

Selection and Select Object

- The `Select` object represents a dropdown list in an HTML form.
- A dropdown list lets a user select one or more options of a limited number of choices.
- For each `<select>` tag in an HTML form, a `Select` object is created.
- You can access a `Select` object by searching through the `elements[]` array of a form, or by using `document.getElementById()`.

Selection and Select Object - Properties

- **`disabled`**
  - Sets or returns whether the dropdown list is disabled, or not.
  - **`element.disabled = true | false`**
  - **`element.disabled`**

```
<script type="text/javascript">

  document.getElementById("mydrop").disabled=true;

</script>
```

Selection and Select Object – Properties (Cont.)

- `form`
  - Returns a reference to the form that contains the dropdown list.
  - `selectObject.form`

```
<script type="text/javascript">

function displayResult() {
    var x=document.getElementById("mySelect").form.id;
    alert(x);
}

</script>
```

Selection and Select Object – Properties (Cont.)

- **`length`**
  - Returns the number of options in a dropdown list.
  - **`selectObject.length = number`**

```
<script type="text/javascript">

 function displayResult() {
    alert(document.getElementById("mySelect").length);
 }

</script>
```

Selection and Select Object – Properties (Cont.)

- **multiple**
  - Sets or returns whether more than one item can be selected from the dropdown list.
  - **selectObject.multiple = true | false**
  - **selectObject.multiple**

```
<script type="text/javascript">

 function displayResult() {
    document.getElementById("mySelect").multiple=true;
 }

</script>
```

Selection and Select Object – Properties (Cont.)

- **`name`**
  - Sets or returns the name of a dropdown list.
  - `selectObject.name = name`
  - `selectObject.name`

Selection and Select Object – Properties (Cont.)

- `selectedIndex`
  - Sets or returns the index of the selected option in a dropdown list.
  - `selectObject.selectedIndex = num`
  - `selectObject.selectedIndex`

Selection and Select Object – Properties (Cont.)

- `size`
  - Sets or returns the number of visible options in a dropdown list.
  - `selectObject.size = integer`
  - `selectObject.size`

Selection and Select Object – Properties (Cont.)

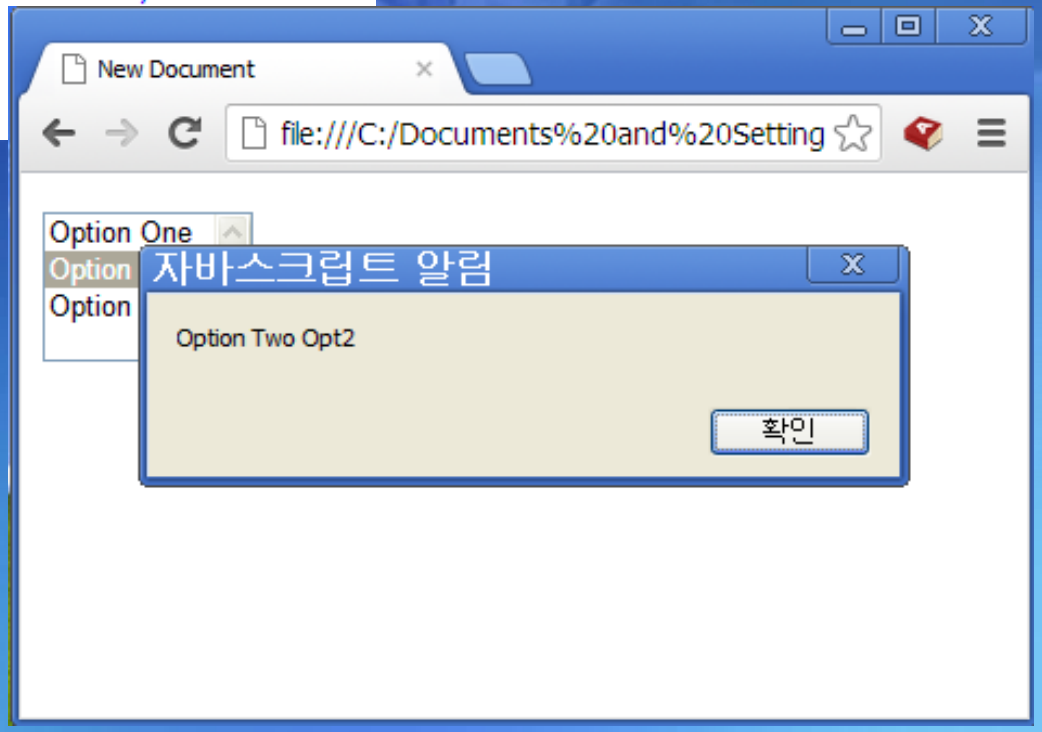
- `type`
  - Returns which type of form element a dropdown list is.
  - For a dropdown list this will be "`select-one`" or "`select-multiple`".
  - `selectObject.type`

Selection and Select Object – Properties (Cont.)

```
<script type="text/javascript">
  function catchEvent(eventObj, event, eventHandler) {
     if (eventObj.addEventListener) {
        eventObj.addEventListener(event, eventHandler,false);
     } else if (eventObj.attachEvent) {
        event = "on" + event;
        eventObj.attachEvent(event, eventHandler);
     }
  }
  catchEvent(window,"load",setupEvents);
  function setupEvents(evnt) {
     catchEvent(document.getElementById("someForm"), "submit", checkForm);
  }
  function checkForm(evnt) {
     var opts = document.getElementById("someForm").selectOpts.options;
     for (var i = 0; i < opts.length; i++) {
        if (opts[i].selected) {
           alert(opts[i].text + " " + opts[i].value);
        }
     }
     return false;
  }
</script>
```

Selection and Select Object – Properties (Cont.)

```
<body>
  <form id="someForm" action="">
  <p>
  <select id="selectOpts" multiple="multiple">
     <option value="Opt1">Option One</option>
     <option value="Opt2">Option Two</option>
     <option value="Opt3">Option Three</option>
  </select>
  <input type="submit" value="Submit" />
  </p>
  </form>
</body>
```



Dynamically Modifying the Selection

- Can create and remove selection list items.
- To add a new option to the application, create a new option element and add it to the `options` array:

```
opts[opts.length] =
    new Option("Option Four", "Opt4");
```

- Because arrays are zero-based, can always add a new array element at the end by using the array's `length` property as the index.

Dynamically Modifying the Selection (Cont.)

- To remove an option, just set the option entry in the array to `null`.
- This resets the array so that there is no gap in the numbering.

```
opts[2] = null;
```

- To remove all options, set the array `length` to zero (`0`):

```
opts.length = 0;
```

Dynamically Modifying the Selection (Cont.)

```
<script type="text/javascript" src="util.js">
  catchEvent(window,"load",setupEvents);

  function setupEvents(evnt) {
      catchEvent(document.getElementById("someForm"), "submit", checkForm);
  }

  function checkForm(evnt) {
      var theEvent = evnt ? evnt : window.event;
      var opts = document.getElementById("someForm").selectOpts.options;

      for (var i = 0; i < opts.length; i++) {
          if (opts[i].selected) {
              opts[i] = null;
          }
      }
      cancelEvent(theEvent);
  }
</script>
```

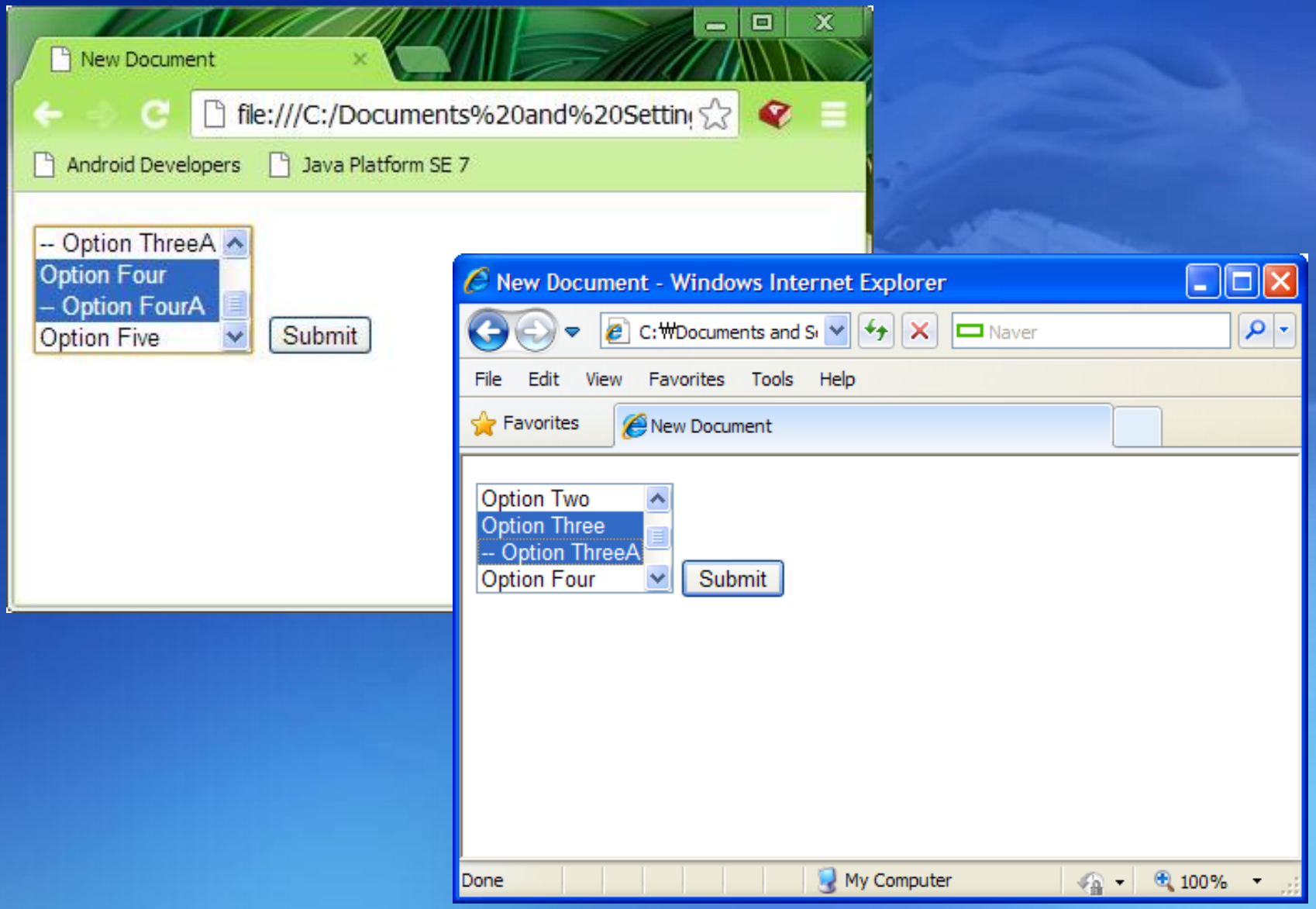
```
<body>
  <form id="someForm" action="">
  <p>
  <select id="selectOpts" multiple="multiple">
      <option value="Opt1">Option One</option>
      <option value="Opt2">Option Two</option>
      <option value="Opt3">Option Three</option>
  </select>
  <input type="submit" value="Submit" />
  </p>
  </form>
</body>
```

Selection and Auto-Selection

```
<script type="text/javascript" src="util.js"></script>
<script type="text/javascript">
 catchEvent(window,"load",setupEvents);
 function setupEvents(evnt) {
    catchEvent(document.getElementById("selectOpts"), "change", checkSelect);
 }
 function checkSelect(evnt) {
    var theEvent = evnt ? evnt : window.event;
    var opts = document.getElementById("someForm").selectOpts.options;
    for (var i = 0; i < opts.length; i++) {
       if (opts[i].selected) {
          switch(opts[i].value) {
             case "Opt1" : opts[i + 1].selected = true;
                           break;
             case "Opt3" : opts[i + 1].selected = true;
                           break;
             case "Opt4" : opts[i + 1].selected = true;
                           break;
          }
       }
    }
    cancelEvent(theEvent);
 }
</script>
```

```
<body>
  <form id="someForm" action="">
  <p>
  <select id="selectOpts" multiple="multiple">
     <option value="Opt1">Option One</option>
     <option value="Opt1a"> -- Option OneA</option>
     <option value="Opt2">Option Two</option>
     <option value="Opt3">Option Three</option>
     <option value="Opt3a"> -- Option ThreeA</option>
     <option value="Opt4">Option Four</option>
     <option value="Opt4a"> -- Option FourA</option>
     <option value="Opt5">Option Five</option>
  </select>
  <input type="submit" value="Submit" />
  </p>
  </form>
</body>
```

Selection and Auto-Selection (Cont.)



Radio Buttons and Checkboxes

- Provide one-click option choosing, usually with a smaller number of options than a selection.
- Differ in that radio buttons allow only one choice, but can select as many checkboxes as you like.

```
<form id="someForm" action="">
 <p>
 <input type="radio" value="Opt 1" name="radiogroup" />Option 1<br />
 <input type="radio" value="Opt 2" name="radiogroup" />Option 2<br />
 </p>
</form>
```

Radio Buttons and Checkboxes (Cont.)

- Notice that the name is the same for both options.
- That's how the buttons are grouped.
- To access the radio button group, use the `document.getElementById` function, passing in the radio button group name:

```
var buttons = document.getElementById("radioGroup");

for (var i = 0; i < buttons.length; i++) {
    if (buttons[i].checked) {
        alert(buttons[i].value);
    }
}
```

Radio Buttons and Checkboxes (Cont.)

- Checkboxes differ that can select more than one checkbox.

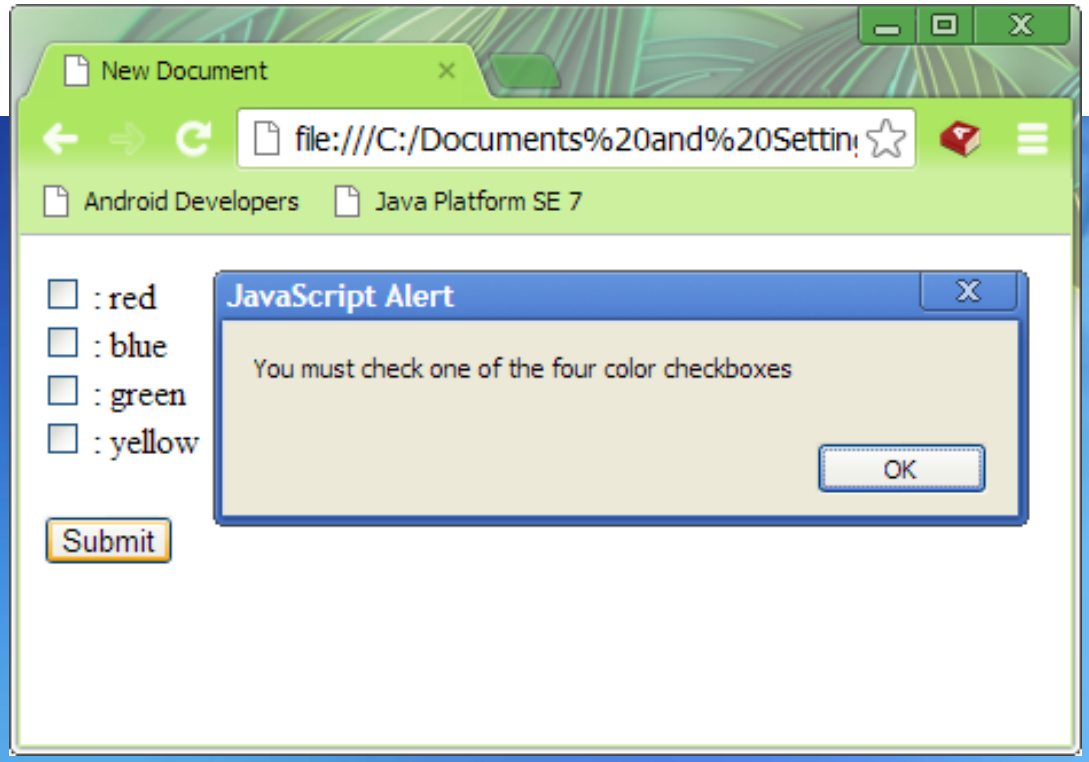
```
<form id="someForm" action="">
   <p>
      Option 1: <input type="checkbox" name="checkbox1" value="Opt1" /><br />
      Option 2: <input type="checkbox" name="checkbox2" value="Opt2" /><br />
   </p>
</form>
```

Radio Buttons and Checkboxes (Cont.)

```
<script type="text/javascript">
  catchEvent(window,"load",setupEvents);
  function setupEvents(evnt) {
     catchEvent(document.getElementById("someForm"), "submit",checkColors);
  }
  function checkColors(evnt) {
     var theEvent = evnt ? evnt : window.event;
     var colorOpts = document.getElementById("someForm").getElementsByTagName("input");
     var isChecked = false;
     for (var i = 0; i < colorOpts.length; i++) {
        if ((colorOpts[i].type == "checkbox") && (colorOpts[i].checked)) {
           isChecked=true;
           break;
        }
     }
     if (!isChecked) {
        alert("You must check one of the four color checkboxes");
        cancelEvent(theEvent);
     }
  }
</script>
```

Radio Buttons and Checkboxes (Cont.)

```
<body>
<form id="someForm" action="">
 <p>
 <input type="checkbox" name="color1" value="red" /> : red<br />
 <input type="checkbox" name="color2" value="blue" /> : blue<br />
 <input type="checkbox" name="color3" value="green" /> : green<br />
 <input type="checkbox" name="color4" value="yellow" /> : yellow<br /><br />
 <input type="submit" value="Submit" />
 </p>
</form>
</body>
```



Radio Buttons and Checkboxes (Cont.)

- Can capture the `clickevent` for the group if you want to modify other form elements based on a radio button or checkbox selection.
- To attach an event handler, you attach it to the group:

```
document.getElementById("someForm").radiogroup.onclick=handleClick;
```

Radio Buttons and Checkboxes (Cont.)

- Can create new checkbox or radio button elements.
- Also attach the new element to the form:

```
var newCheckbox = document.createElement("input");
newCheckbox.type="checkbox";
newCheckbox.name="colors1";
newCheckbox.checked=true;
someForm.appendChild(newCheckBox);
```

The text, textarea, password, and hidden Input Elements

- Define the single-row text-based input elements in HTML as follows:

```
<input type="text|hidden|password"
name="fieldName" value="Some value" />
```

- The textarea field is similar.
- Except that unlike the other input fields, it has an opening and closing tag.
- Can set both column and row widths.

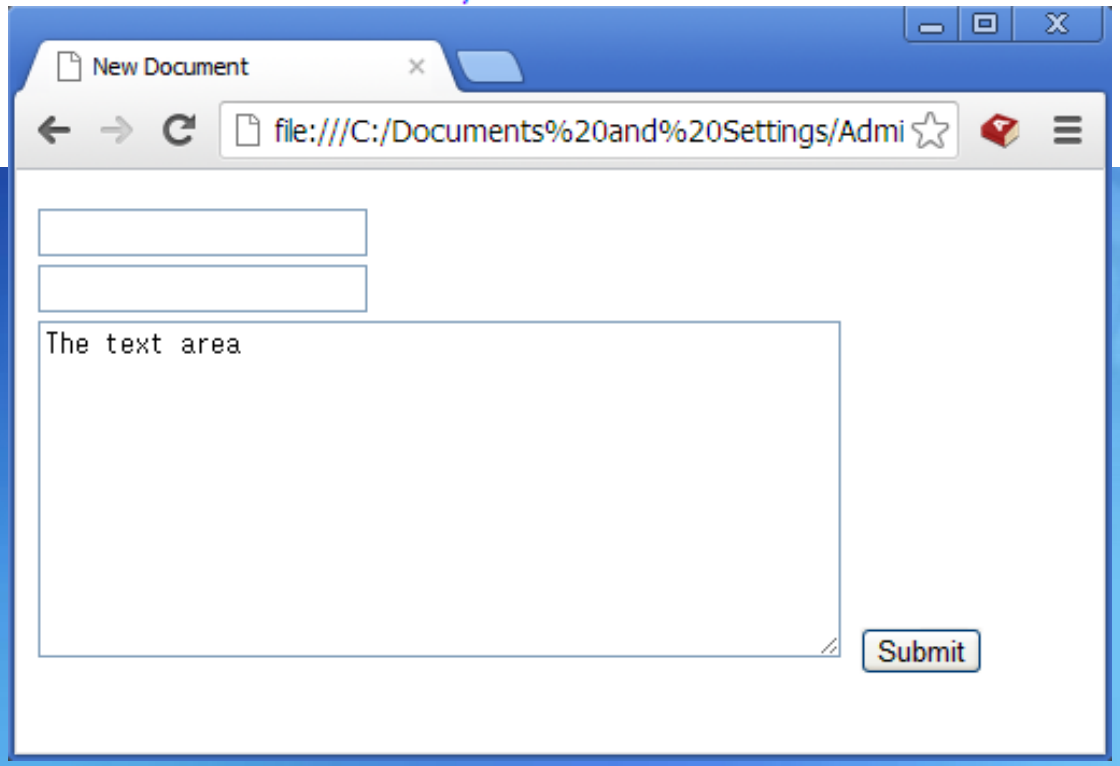
```
<textarea name="fieldName" rows="10"
  cols="10">Initial text</textarea>
```

The text, textarea, password, and hidden Input Elements (Cont.)

```
<script type="text/javascript">
  catchEvent(window,"load",setupEvents);
  function setupEvents(evnt) {
      catchEvent(document.getElementById("someForm"), "submit",validateForm);
  }
  function validateForm(evnt) {
      var theEvent = evnt ? evnt : window.event;
      var strResults = "";
      var textInputs =
      document.getElementById("someForm").getElementsByTagName("input");
      for (var i = 0; i < textInputs.length; i++) {
          if (textInputs[i].type != "submit") {
              strResults += textInputs[i].value;
          }
      }
      document.getElementById("text4").value=strResults;
      cancelEvent(theEvent);
  }
</script
```

The text, textarea, password, and hidden Input Elements (Cont.)

```
<body>
  <form id="someForm" action="">
    <p>
    <input type="text" name="text1" /><br />
    <input type="password" name="text2" /><br />
    <input type="hidden" name="text3" value="hidden value" />
    <textarea name="text4" cols="50" rows="10">The text area</textarea>
    <input type="submit" value="Submit" />
    </p>
  </form>
</body>
```



Text Validation

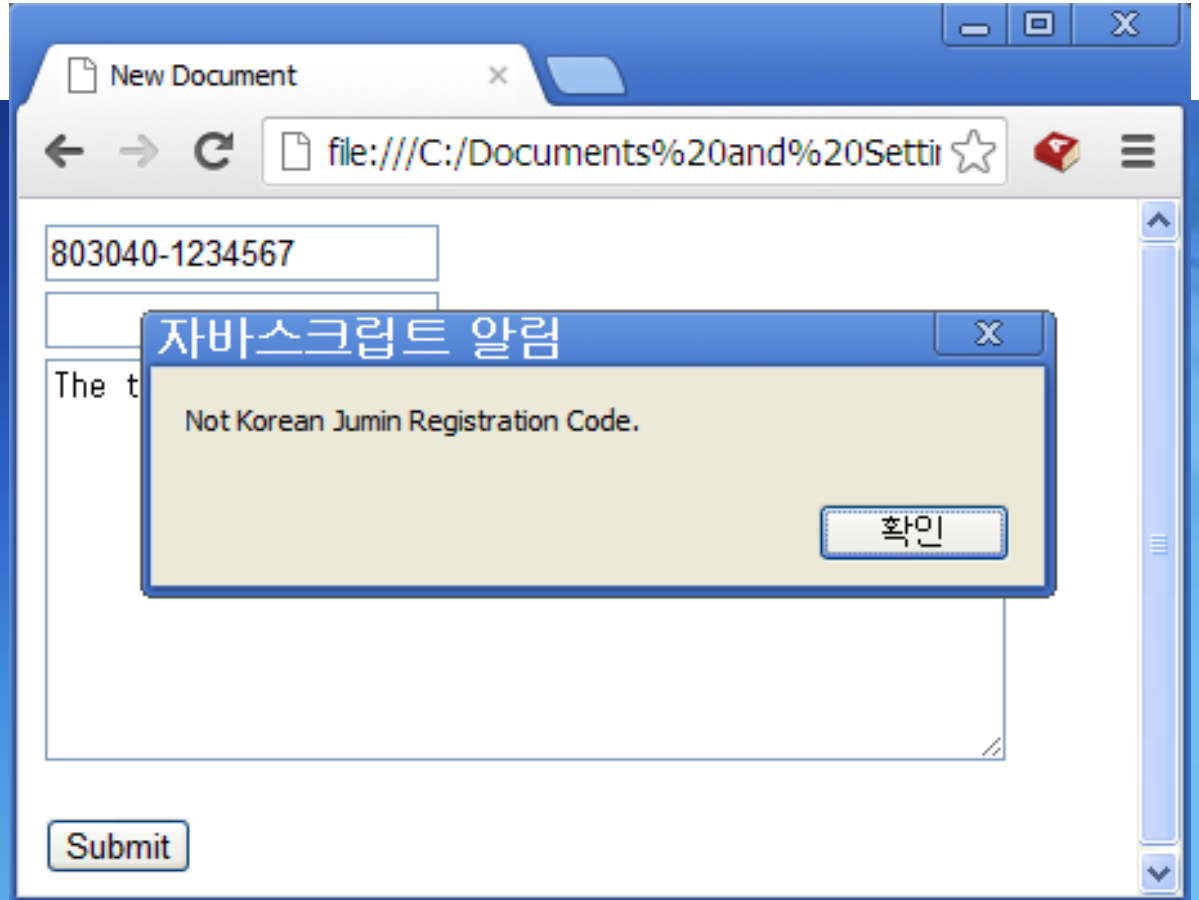
- When the cursor moves into a text input field, a `focus` event is fired.
- When the cursor leaves, the `blur` event is fired.
- A `change` event happens when the contents of the field are changed and the cursor moves out of the field.
- Capture the `change` event to validate contents.
- The `blur` event to make sure the field has some value.

Text Validation (Cont.)

```
catchEvent(window,"load",setupEvents);
function setupEvents(evnt) {
    catchEvent(document.getElementById("text2"), "blur",checkRequired);
    catchEvent(document.getElementById("text1"), "change", validateField);
}
function checkRequired (evnt) {
    var theEvent = evnt ? evnt : window.event;
    var target = theEvent.target ? theEvent.target : theEvent.srcElement;
    var txtInput = target.value;
    if (txtInput == null || txtInput == "") {
        alert("value is required in field");
    }
}
function validateField(evnt) {
    var theEvent = evnt ? evnt : window.event;
    var target = theEvent.target ? theEvent.target : theEvent.srcElement;
    var rgEx = /^\d{2}[0-1]\d[0-3]\d-[1-4]\d{6}$/g;
    var OK = rgEx.exec(target.value);
    if (!OK) {
        alert("Not Korean Jumin Registration Code.");
    }
}
```

Text Validation (Cont.)

```
<body>
  <form name="someForm">
    <input type="text" name="text1" id="text1" /><br />
    <input type="password" name="text2" id="text2" /><br />
    <input type="hidden" name="text3" value="hidden value" />
    <textarea name="text4" cols=50 rows=10>The text area</textarea><br /><br />
    <input type="submit" value="Submit" />
  </form>
</body>
```



Input Fields and Regular Expression Validation

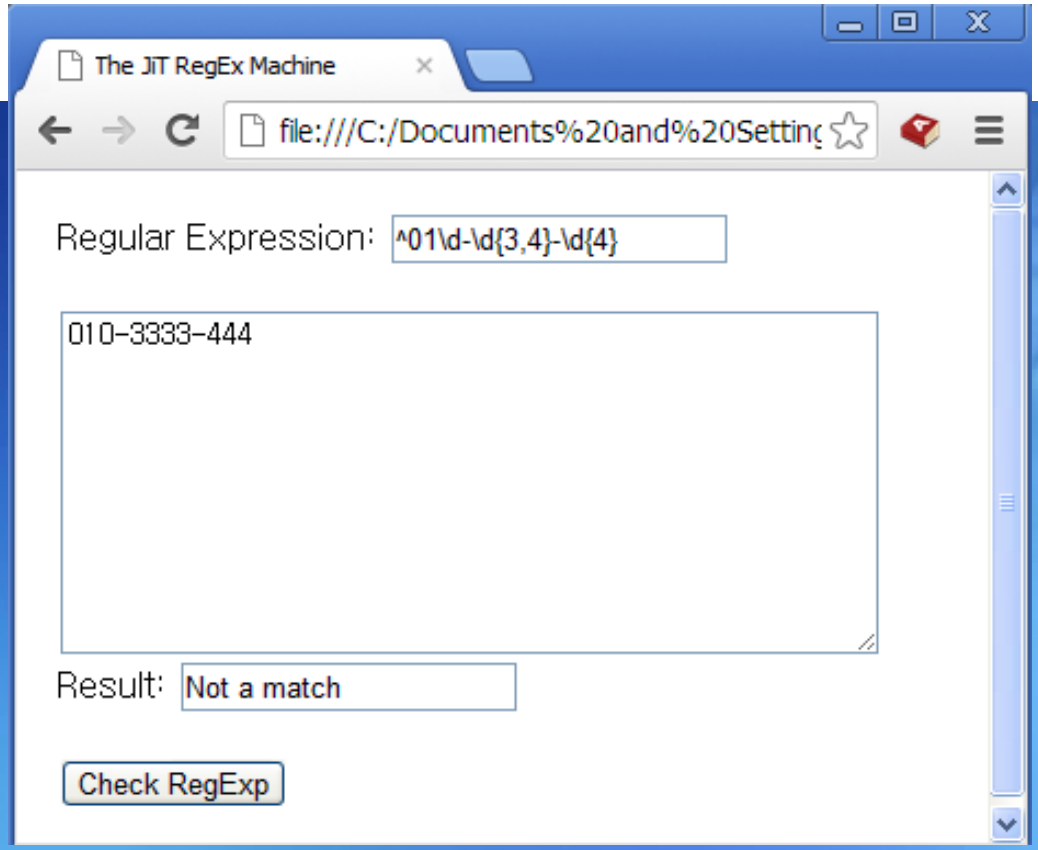
- Certain types of fields may require a specific format.
- Warranty or purchase certificates
- Email addresses
- Phone numbers
- SSNs or other forms of identification
- Dates
- State abbreviations
- Credit card numbers
- Web page URLs or other forms of Uniform Resource Identifiers (URIs)

Input Fields and Regular Expression Validation (Cont.)

```
<script type="text/javascript" src="util.js"></script>
<script type="text/javascript">
  catchEvent(window, 'load', setupEvents);
  function setupEvents(evnt) {
      document.someForm.onsubmit=validateField;
  }
  function validateField(evnt) {
      var rgEx = new RegExp(document.someForm.text1.value);
      var OK = rgEx.exec(document.someForm.text2.value);
      if (!OK) {
          document.someForm.text3.value = "Not a match";
      } else {
          document.someForm.text3.value = "The Pattern matched!";
      }
      return false
  }
</script>
```

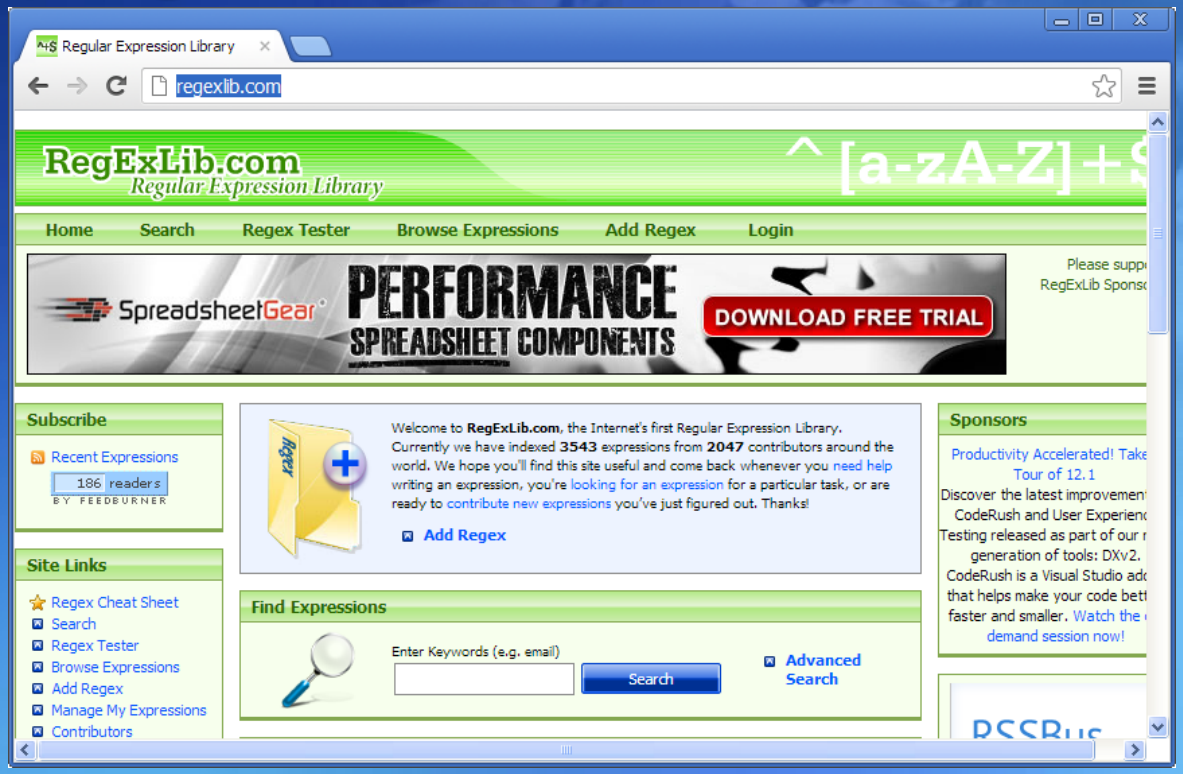
Input Fields and Regular Expression Validation (Cont.)

```
<body>
  <form name="someForm" style="padding: 10px">
     Regular Expression: <input type="text" name="text1" /><br /><br />
     <textarea name="text2" cols=50 rows=10></textarea><br />
     Result: <input type="text" name="text3" /><br /><br />
     <input type="submit" value="Check RegExp" />
  </form>
 </body>
```



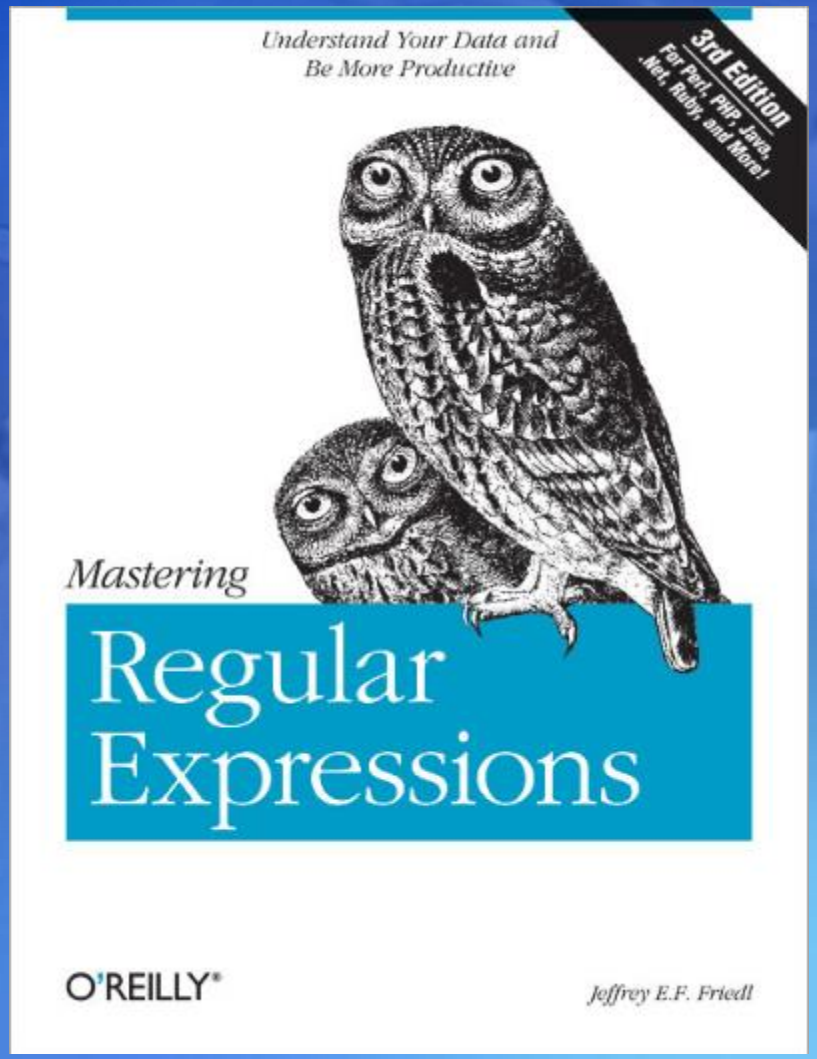
Input Fields and Regular Expression Validation (Cont.)

- For more information of regular expression, visit to Regular Expression Library, an invaluable resource located at http://regexlib.com



Input Fields and Regular Expression Validation (Cont.)

- Can buy a copy of Mastering Regular Expression by Jeffrey E.F. Friedl(O'Reilly).



Test Your Knowledge : Quiz

1. How do you stop a form submittal if the form data is incomplete or invalid?

Test Your Knowledge : Quiz

1. How do you stop a form submittal if the form data is incomplete or invalid?

If you're using DOM Level 0 events, returning `false` from the event handler.
If you're using DOM Level 2, set `cancelBubble` to `true` for IE, and call the `preventDefault` method for other browsers.

Test Your Knowledge : Quiz (Cont.)

2. What event(s) do you want to capture on text input fields to validate the contents before the form is submitted?

Test Your Knowledge : Quiz (Cont.)

2. What event(s) do you want to capture on text input fields to validate the contents before the form is submitted?

The `blur` event is triggered when the field loses focus.

Test Your Knowledge : Quiz (Cont.)

3. What code would you use to ensure that a field has only letters and whitespace?

Test Your Knowledge : Quiz (Cont.)

3. What code would you use to ensure that a field has only letters and whitespace?

```
var fieldPattern = /^[A-Za-z\s]*$/g;
var OK =
    fieldPattern.exec(document.forms[0].text1.value);
```

Test Your Knowledge : Quiz (Cont.)

4. Create the JavaScript that captures an event when a radio button is checked, then disables a text input field if one button is clicked, and enables it if another button is clicked.

Test Your Knowledge : Quiz (Cont.)

4. Create the JavaScript that captures an event when a radio button is checked, then disables a text input field if one button is clicked, and enables it if another button is clicked.

```
document.forms[0].radiogroup[0].onclick=handleClick;
document.forms[0].radiogroup[1].onclick=handleClick;
```

```
function handleClick() {
      if (document.forms[0].radiogroup[1].checked) {
            document.forms[0].submit.disabled=true;
      } else {
            document.forms[0].submit.disabled=false;
      }
}
```