Intro to JavaScript Events

JavaScript Events

- Events in JavaScript let a web page react to some type of input
- Many different ways to handle events due to history/vendor differences but we have a generally standard way now
  - Will not cover all event levels
- Events
  - W3C DOM Standard; attach to HTML elements
  - DOM Levels 0 to 3

Event Handlers to HTML Attributes

- We can add an event handler to a HTML element
  - onkeydown, onkeyup, onkeypress
  - onclick, onmouseover, onmouseout, onmousedown, onmouseup
  - Others…

```
<form name="theForm" action="">
<input type=text name="myTextBox" id="text1" value="1" onclick="addOne();">
</form>

<div id="myDiv" onmouseover="changeColor(this,'red');"
        onmouseout="changeColor(this,'black');">
Hello there
</div>
```

```
<script type="text/javascript">
function addOne()
{
        var el = document.theForm.myTextBox;
        el.value = parseInt(el.value) + 1;
}

function changeColor(el, col)
{
 el.style.color = col;
}
</script>
```

DOM Event Handlers

- HTML element event handlers fine for elements, not good for the entire webpage if it means adding handlers to every single element
- Various DOM Event Handlers
  - Complicated by different methods for different browsers and different versions

```
<form name="theForm" action="">
<input type=button name="myButton" id="theButton" value="Click Me">
<input type=text name="myTextBox" id="text1" value="1">
</form>
```

```
<script type="text/javascript">
var btn = document.theForm.myButton;
if (typeof addEventListener === "function")
{
        btn.addEventListener("click", addOne, false); // Compliant browsers
}
else
{
        btn.attachEvent("onclick", addOne);   // IE8 and lower
}

function addOne()
{
        var el = document.theForm.myTextBox;
        el.value = parseInt(el.value) + 1;
}
</script>
```

Evolution – Make a event utility

In file eventutility.js:

```
var eventUtility = {
        addEvent : (function() {
                if (typeof addEventListener === "function") {
                        return function(obj, evt, fn) {
                                obj.addEventListener(evt, fn, false);
                        };
                } else {
                        return function(obj, evt, fn) {
                                obj.attachEvent("on" + evt, fn);
                        };
                }
        }()),
        removeEvent : (function() {
                if (typeof addEventListener === "function") {
                        return function(obj, evt, fn) {
                                obj.removeEventListener(evt, fn, false);
                        };
                } else {
                        return function(obj, evt, fn) {
                                obj.detachEvent("on" + evt, fn);
                        };
                }
        }())
};
```

HTML / JavaScript Code

```
<form name="theForm" action="">
<input type=button name="myButton" id="theButton" value="Click Me">
<input type=text name="myTextBox" id="text1" value="1">
</form>

<script type="text/javascript" src="eventutility.js"></script>
<script type="text/javascript">
var btn = document.theForm.myButton;
eventUtility.addEvent(btn, "click", addOne);

function addOne()
{
        var el = document.theForm.myTextBox;
        el.value = parseInt(el.value) + 1;
}
</script>
```

Accessing the Event Target

- Use event.type and event.target to determine the event and target of the event

```
<form name="theForm" action="">
<input type=button name="myButton" id="theButton" value="Click Me">
<input type=text name="myTextBox" id="text1" value="1">
</form>

<script type="text/javascript" src="eventutility.js"></script>
<script type="text/javascript">
var btn = document.theForm.myButton;
eventUtility.addEvent(btn, "click", eventHandler);

function eventHandler(event)
{
        alert(event.type);
        event.target.style.backgroundColor = "green";
}
</script>
```

Event Target

- Can use the same event handler for multiple targets

```
<form name="theForm" action="">
<input type=button name="myButton" id="theButton" value="Click Me">
<input type=text name="mainTextBox" id="text0" value="" width=20><br/>
<input type=text name="myTextBox" id="text1" value="1">
</form>


<script type="text/javascript" src="eventutility.js"></script>
<script type="text/javascript">
var btn = document.theForm.myButton;
eventUtility.addEvent(btn, "click", eventHandler);

var txt = document.theForm.text0;
eventUtility.addEvent(txt, "keypress", eventHandler);

function eventHandler(event)
{
    if (event.type == "keypress") {
        document.theForm.myTextBox.value = parseInt(document.theForm.myTextBox.value) + 1;
    }
    else if (event.type == "click") {
        alert("You clicked on " + event.target.id);
    }
}
</script>
```

AJAX

- Term invented by Jesse Garrett, 2005, "Ajax: A New Approach to Web Applications"
  - Asynchronous JavaScript + XML
  - Although XML still used, other formats also used as well
- In general, the use of JavaScript to send and receive data using HTTP without reloading the page
  - Allows for dynamic pages without clunky submit/reload paradigm

AJAX and the XHR Object

- XHR = XMLHttpRequest Object
  - Originated as a component, XmlHttp, in Microsoft's MSXML Library
    - Still necessary if you're programming for old versions of IE
    - Built into modern browsers
  - Despite the XML name you can retrieve more than XML and is commonly used with plaintext
  - Must be used with a HTTP server
- Creating an XHR object:

```
var xhr = new XMLHttpRequest();
```

XHR Object

- Call the open method
  xhr.open("GET", "info.txt", true);

  Asynchronously retrieve “info.txt” from the same website/port, can also use POST
- Five ready states
  - 0: Object created but not initialized
  - 1: Object initialized but request not sent
  - 2: Request sent
  - 3: Response received from HTTP server
  - 4: Requested data fully received
- Same response codes as HTTP
  - 2xx = Success, 4xx = Client Error, 5xx = Server Error

Sample XHR Code

```
<html>
<header>
<title>This is a test</title>
</header>

<body>
<H1>Testing</H1>

<script>
var xhr = new XMLHttpRequest();
xhr.open("GET", "info.txt", true);

xhr.onreadystatechange = function() {
    if (xhr.readyState == 4) {
        alert("response received: " + xhr.responseText);
    }
}
xhr.send(null);

</script>

</body>
</html>
```

Ajax POST

- Use POST requests to send data and receive a response; couple with events for dynamic page

```
<form name="theForm" action="">
<input type=text name="myTextBox" id="text1" value=""><br/>
<div id="suggestion">Suggestion goes here from server</div>
</form>

<script type="text/javascript" src="eventutility.js"></script>
<script type="text/javascript">
var xhr = new XMLHttpRequest();
xhr.onreadystatechange = function() {
      if (xhr.readyState == 4) {
            processResponse(xhr.responseText);
      }
}
```

```
var txt = document.theForm.text1;
eventUtility.addEvent(txt, "keyup", eventHandler);

// When we press a key send to the server like it is a <form>
// and wait for a response
function eventHandler(event)
{
 var data = "myTextBox=" +
        encodeURIComponent(document.theForm.myTextBox.value) +
        "&otherParameter=someValue";
 xhr.open("POST", "ajax_test.php");
 xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
 xhr.send(data);
}

// Display response from server in DIV
function processResponse(responseData)
{
 var el = document.getElementById("suggestion");
 el.innerHTML = "<B>" + responseData + "</B>";
}
</script>
```

Ajax Server Side PHP

```
<?php
 if (isset($_REQUEST['myTextBox']))
 {
      // Normally you would do some more interesting lookup than this
      // canned example
      $txt= strtolower($_REQUEST['myTextBox']);
      if (strlen($txt)>0)
      {
       $firstLetter = $txt[0];
       if ($firstLetter == 'a')
            print "Alfred";
       else if ($firstLetter == 'k')
            print "Kenrick";
       else if ($firstLetter == 'b')
            print "Bob";
       else if ($firstLetter == 'j')
            print "Jose";
      }
      else
            print "";
 }
 else
   print "";
?>
```