HTML DOM

Sang Shin
JPassion.com
“Code with Passion!”

Topics

- HTML DOM objects
- Window object
- Document object
- DOM APIs
- DOM Event handling
- Form handling
- Event object

HTML DOM Objects

HTML DOM Objects

- The HTML DOM defines a standard set of objects for HTML, and a standard way to access and manipulate HTML elements
  - All HTML elements, along with their containing text and attributes, can be accessed through the DOM
  - The contents can be modified or deleted, and new elements can be created.
- The HTML DOM is platform and language independent
  - It can be used by any programming language like Java, JavaScript, and VBScript

HTML DOM Objects (1)

- Anchor object
- Document object*
- Event object*
- Form and Form Input object*
- Frame, Frameset, and IFrame objects
- Image object
- Location object
- Navigator object

HTML DOM Objects (1)

- Anchor object
- Document object*
- Event object*
- Form and Form Input object*
- Frame, Frameset, and IFrame objects
- Image object
- Location object
- Navigator object

We are going to focus DOM objects with *

HTML DOM Objects (2)

- Option and Select objects
- Screen object
- Table, TableHeader, TableRow, TableData objects
- Window object*

HTML DOM Objects:
Window Object

Window Object

- Represents browser window
  - This is the global JavaScript object in a browser, in which all global variables and functions are defined as its properties
  - Can be accessed through “Window”, “window” or “this” in global scope
- Properties of Window object
  - document – You can use *document.write(“hello”); or window.document.write(“hello”);*
  - innerHeight, innerWidth
  - ...
- Methods of Window object (Properties of Window object whose value are functions)
  - alert(“my message”), prompt(“message”, “defaultmessage”)
  - resizeBY(X, Y), close()
  - ...

Window: User Interaction Methods

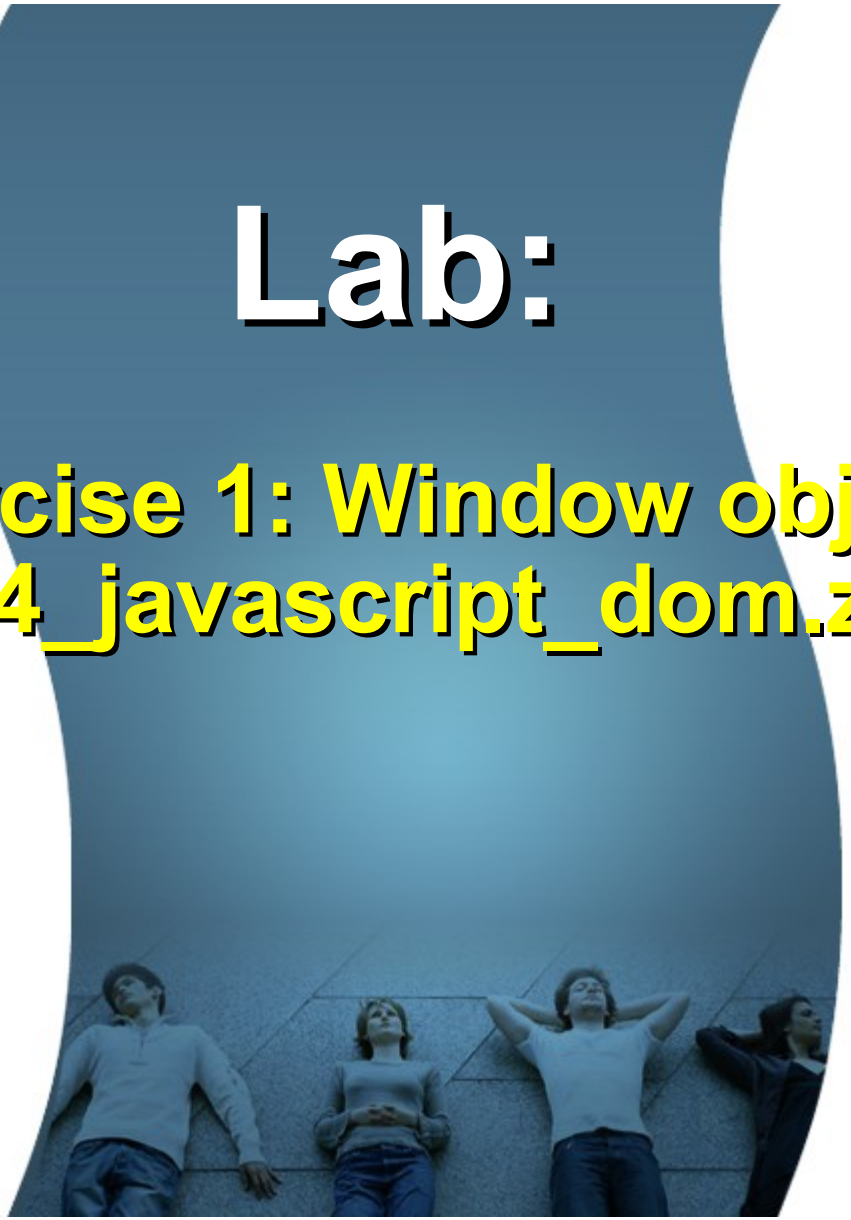
- Alert box
  - User will have to click "OK" to proceed
  - alert("sometext")
- Confirm box
  - User will have to click either "OK" or "Cancel" to proceed
  - confirm("sometext")
- Prompt box
  - User will have to click either "OK" or "Cancel" to proceed after entering an input value
  - prompt("sometext","defaultvalue")

Window Object's Methods and Properties

```
// "this" in global scope represents Window object
console.log("this: " + this); // Window

// User interaction methods of Window object
this.alert("sometext"); // same as alert("sometext");
confirm("sometext"); // same as this.confirm("sometext");
prompt("sometext", "Live your life with Passion!");

// Access document object through document property of Window object
this.document.write("Helloworld!"); // same as document.write("Helloworld!");
```

Lab:

Exercise 1: Window object
4264_javascript_dom.zip



HTML DOM Objects:
Document Object

Document Object

- Represents HTML Document
  - Container of HTML HEAD and BODY objects
- Methods
  - write(“Helloworld!”);
  - getElementById(<id>), getElementByName(<name>)
- Properties
  - anchors
  - charset
  - childNodes
  - ...

Write text to the output

```
<html>
<body>

<script type="text/javascript">
    document.write("Hello World!")
</script>

</body>
</html>
```

Write text with Formatting to the output

```
<html>
<body>

<script type="text/javascript">
    document.write("<h1>Hello World!</h1>")
</script>

</body>
</html>
```

getElementById("myid") - accesses the element with the specified id

```
<html>

<head>
<script type="text/javascript">
    function getElement() {
        var x=document.getElementById("myHeader")
        alert("I am a " + x.tagName + " element")
    }
</script>
</head>

<body>
<h1 id="myHeader" onclick="getElement()">Click to see what element I am!</h1>
</body>

</html>
```

getElementsByName("..") - accesses all elements with the specified name

```
<html>
<head>
<script type="text/javascript">
    function getElements() {
        var x=document.getElementsByName("myInput")
        alert(x.length + " elements!")
    }
</script>
</head>

<body>
<input name="myInput" type="text" size="20"><br />
<input name="myInput" type="text" size="20"><br />
<input name="myInput" type="text" size="20"><br />
<br />
<input type="button" onclick="getElements()" value="How many elements named
    'myInput'?">
</body>
</html>
```

Return the innerHTML of the first anchor in a document

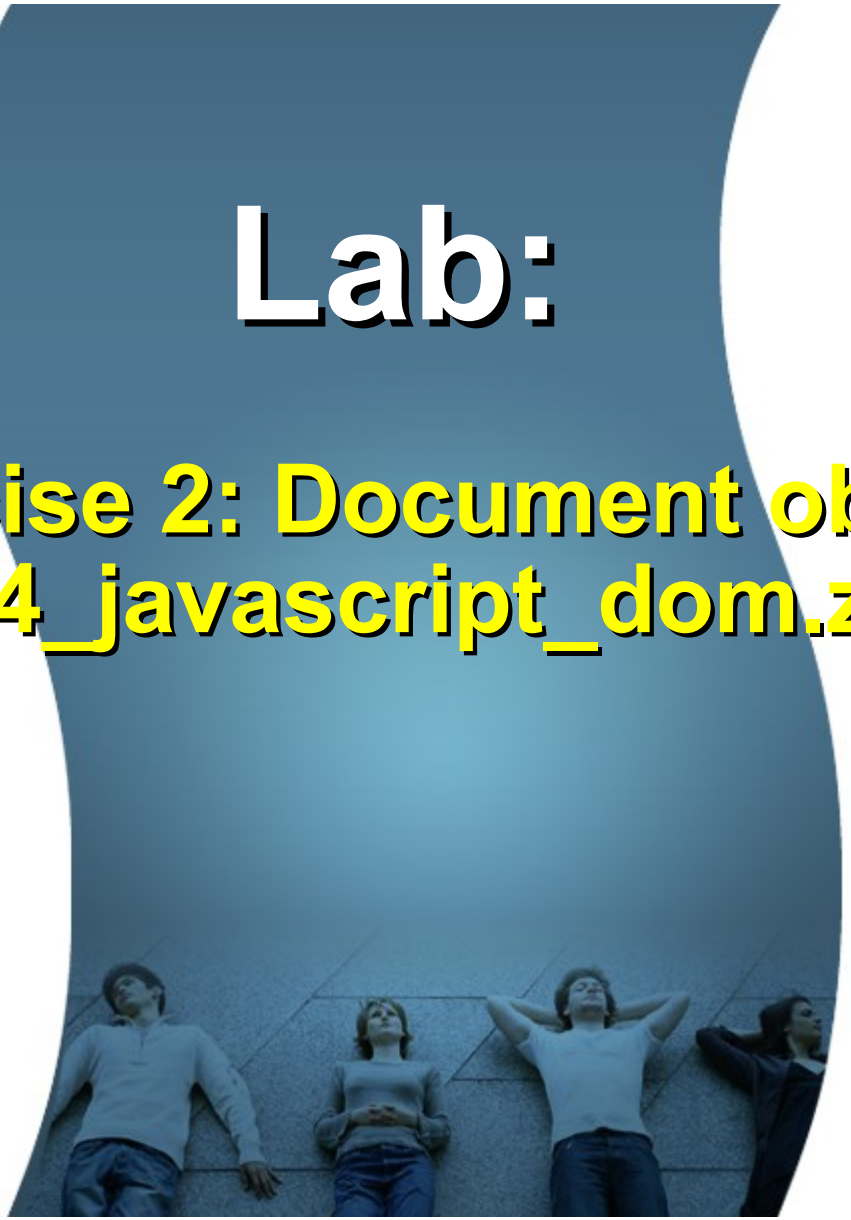
```
<html>
<body>

<a name="first">First anchor</a><br />
<a name="second">Second anchor</a><br />
<a name="third">Third anchor</a><br />
<br />

InnerHTML of the first anchor in this document:
<script type="text/javascript">
    document.write(document.anchors[0].innerHTML); // First anchor
</script>

</body>

</html>
```

Access an item in a collection

```
<html>
<body>
  <form id="Form1" name="Form1">
    Your name: <input type="text">
  </form>
  <form id="Form2" name="Form2">
    Your car: <input type="text">
  </form>

<p>
To access an item in a collection you can either use the number or the name of the item:
</p>

<script type="text/javascript">
document.write("<p>The first form's name is: " + document.forms[0].name + "</p>")
document.write("<p>The first form's name is: " + document.getElementById("Form1").name + "</p>")
</script>

</body>
</html>
```

Lab:

Exercise 2: Document object 4264_javascript_dom.zip



DOM APIs

DOM API Examples

- document.getElementById("myId")
- document.getElementsByName("myInput")
- document.getElementsByClassName("myClass")
- document.getElementsByTagName("li")

- document.getElementById("myId").firstChild.nextSibling
- document.getElementById("myId").lastChild.innerHTML

JavaScript libraries such as jQuery, Dojo, Prototype, etc. provide “easier to use” APIs than DOM APIs for element access, manipulation, and event handling.

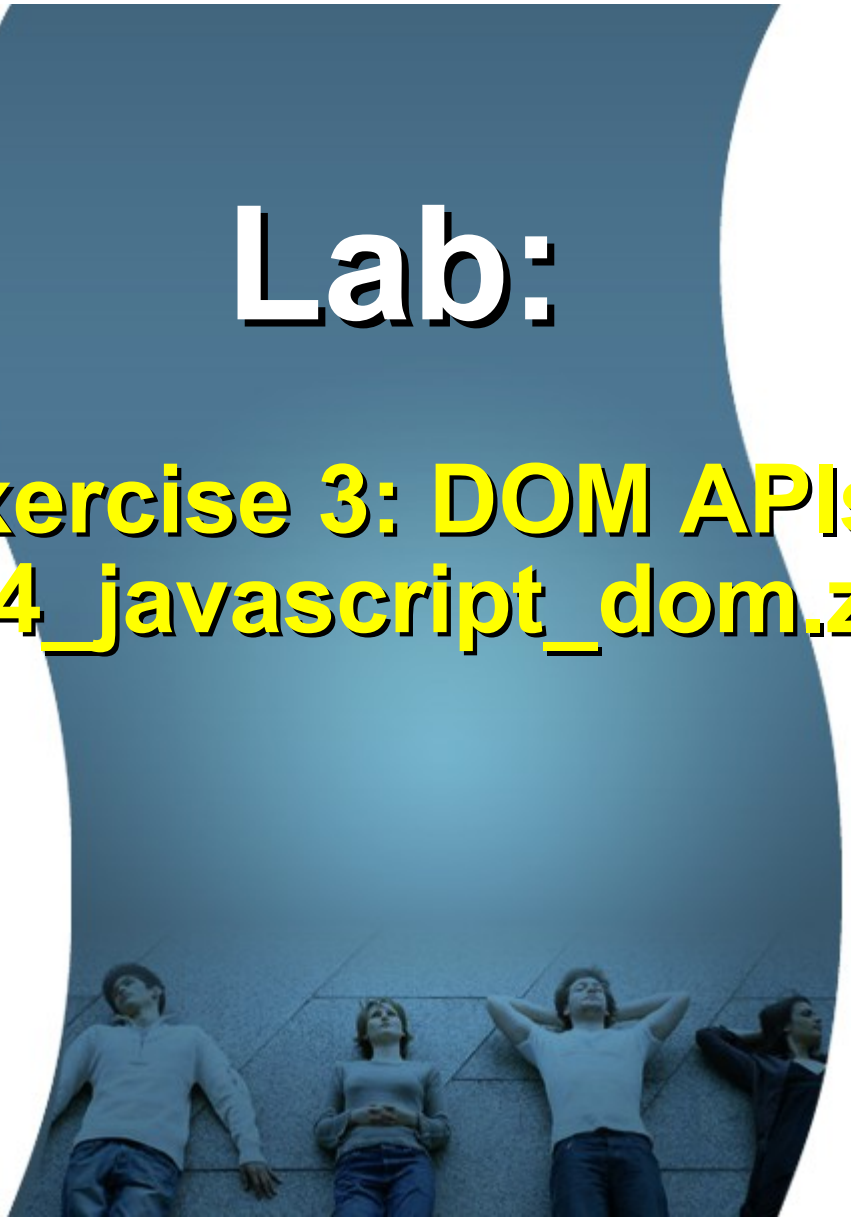
innerHTML

- Each HTML element has an innerHTML property that defines both the HTML code and the text that occurs between that element's opening and closing tag

```
mygreeting_var = document.getElementById('mygreeting_id');
mygreeting_var.innerHTML = 'Good Day!';
```

Lab:

Exercise 3: DOM APIs
4264_javascript_dom.zip



DOM Event Handling

Events & Event Handlers

- Every element on a web page has certain events which can trigger invocation of event handlers
- Attributes are inserted into HTML tags to define events and event handlers
- Examples of events
  - A mouse click
  - A web page or an image loading
  - Mousing over a hot spot on the web page
  - Selecting an input box in an HTML form
  - Submitting an HTML form
  - A keystroke

Events (1)

- onabort - Loading of an image is interrupted
- onblur - An element loses focus
- onchange - The content of a field changes
- onclick - Mouse clicks an object
- ondblclick - Mouse double-clicks an object
- onerror - An error occurs when loading a document or an image
- onfocus - An element gets focus
- onkeydown - A keyboard key is pressed

We are going to focus on these blue-colored events

Events (2)

- onkeypress - A keyboard key is pressed or held down
- onkeyup - A keyboard key is released
- onload - A page or an image is finished loading
- onmousedown - A mouse button is pressed
- onmousemove - The mouse is moved
- onmouseout - The mouse is moved off an element
- onmouseover - The mouse is moved over an element
- onmouseup - A mouse button is released

Events (3)

- onreset - The reset button is clicked
- onresize - A window or frame is resized
- onselect - Text is selected
- onsubmit - The submit button is clicked
- onunload - The user exits the page

onload & onUnload Events

- The *onload* and *onUnload* events are triggered when the user enters or leaves the page
- The *onload* event is often used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information

onFocus, onBlur and onChange

- The *onFocus*, *onBlur* and *onChange* events are often used in combination with validation of form fields.
- Example: The *checkEmail()* function will be called whenever the user changes the content of the field:

```
<input type="text" size="30"
        id="email" onchange="checkEmail()">;
```

Example & Demo: onblur

```
<html>
<head>
<script type="text/javascript">
    function upperCase() {
        var x=document.getElementById("fname").value
        document.getElementById("fname").value=x.toUpperCase()
    }
</script>
</head>

<body>

Enter your name:
<input type="text" id="fname" onblur="upperCase()">

</body>
</html>
```

onMouseOver and onMouseOut

- *onMouseOver* and *onMouseOut* are often used to create "animated" buttons.
- Example: An alert box appears when an *onMouseOver* event is detected, “return false” prevents the default event from occuring (prevents going to the website in the example below)

```
<a href="http://www.jpassion.com"
    onmouseover="alert('An onMouseOver event');">
    <img src="duke.gif" width="100" height="30">
</a>
```

Lab:

Exercise 4: DOM Event handling
4264_javascript_dom.zip

Form Handling

onsubmit

- The *onsubmit* event occurs when the submit button in a form is clicked.

```
<form name="myform" action="dummy.html" onsubmit="greeting()">
    <input type="text" name="myname" />
   <input type="submit" value="Submit" />
</form>
```

onsubmit with validation

- The *onSubmit* event is used to validate all form fields before submitting it.
- Example: The *checkForm()* function will be called when the user clicks the submit button in the form. If the field values are not accepted, the submit should be canceled. The function *checkForm()* returns either true or false. If it returns true the form will be submitted, otherwise the submit will be canceled:

```
<form method="post" action="dummy.html"
        onsubmit="checkForm()">
```

onSubmit with validation

```
<html>
<head>
<script type="text/javascript">
   function validate() {
      //  return true or false based on validation logic
   }
</script>
</head>

<body>
      <form action="tryjs_submitpage.htm" onsubmit="validate()">
         Name (max 10 chararcters): <input type="text" id="fname" size="20"><br />
         Age (from 1 to 100): <input type="text" id="age" size="20"><br />
         E-mail: <input type="text" id="email" size="20"><br />
         <br />
         <input type="submit" value="Submit">
      </form>
</body>
</html>
```

Form Handling: Validation with Regular Expression

Regular Expression

- A regular expression is an object that describes a pattern of characters.
- Regular expressions are useful for performing pattern-matching input form validation
- There are two ways for creating regular expression
  - Using literal syntax

    var myRegExp = /mypattern/;
  - Using RegExp() constructor

    var myRegExp = new RegExp("mypattern");

Regular Expression Methods

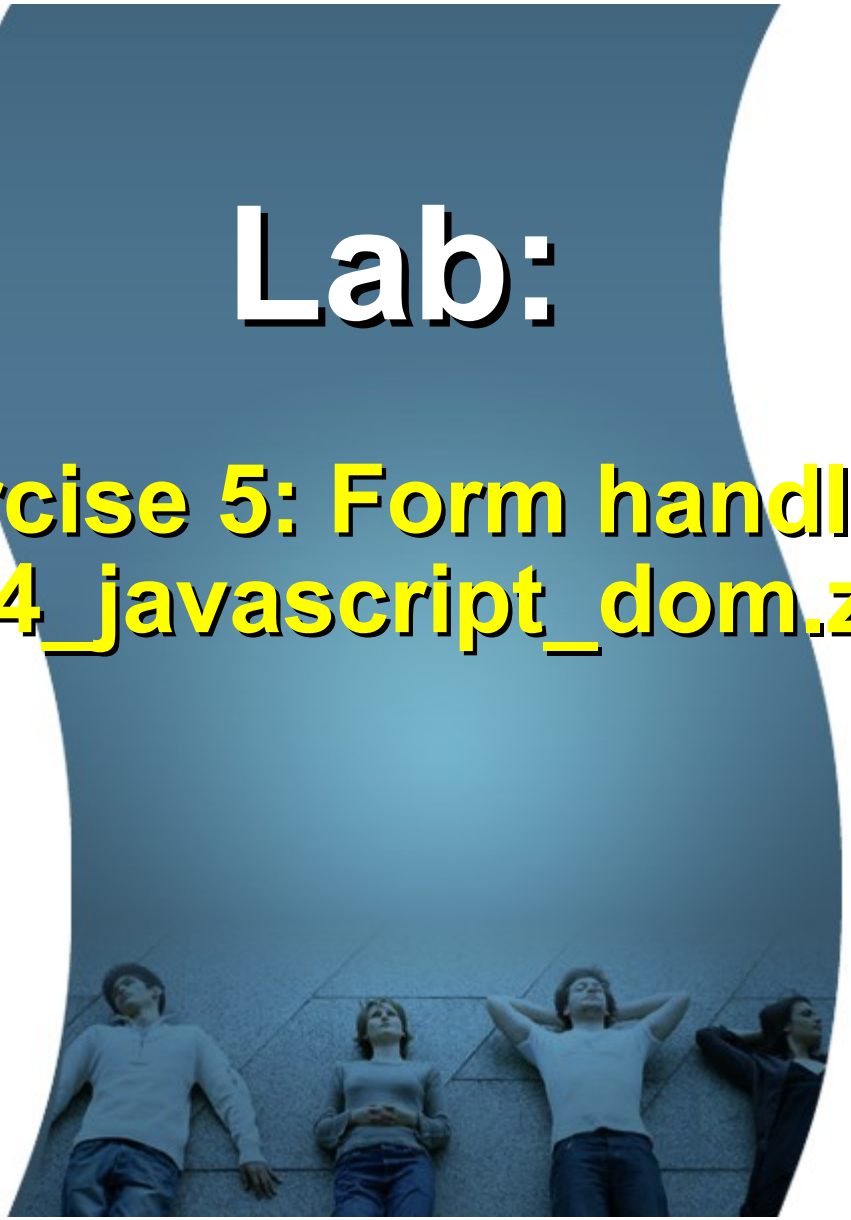
- myRegExp.test() method
  - The test() method takes one argument, a string, and checks whether that string contains a match of the pattern specified by the regular expression.
  - It returns true if it does contain a match and false otherwise
- myRegExp.exec() method
  - The exec() method takes one argument, a string, and checks whether that string contains one or more matches of the pattern specified by the regular expression
  - If one or more matches is found, the method returns a result array with the starting points of the matches.
  - If no match is found, the method returns null

Validation with RegExp

```
<script type="text/javascript">
// A questionmark (?) indicates that the preceding character
// should appear zero or one times in the pattern
var SSN_RegExp = /^[0-9]{3}[\- ]?[0-9]{2}[\- ]?[0-9]{4}$/;

// Inspect RegExp object
console.dir(SSN_RegExp);

function checkSSN(ssn) {
    if (SSN_RegExp.test(ssn)) {
        alert("You entered an valid SSN");
    } else {
        alert("You entered an invalid SSN.");
    }
}

</script>
```

Lab:

Exercise 5: Form handling
4264_javascript_dom.zip



Event Object

Event Object: What are the coordinates of the cursor?

```
<html>
<head>
<script type="text/javascript">
    function show_coords(event) {
        x=event.clientX
        y=event.clientY
        alert("X coords: " + x + ", Y coords: " + y)
    }
</script>
</head>

<body onmousedown="show_coords(event)">
<p>Click in the document. An alert box will alert the x and y coordinates of the
    cursor.</p>
</body>

</html>
```

Event Object: What is the unicode of the key pressed?

```
<html>
<head>
<script type="text/javascript">
    function whichButton(event) {
        alert(event.keyCode)
    }

</script>
</head>

<body onkeyup="whichButton(event)">
<p><b>Note:</b> Make sure the right frame has focus when trying this example!</p>
<p>Press a key on your keyboard. An alert box will alert the unicode of the key
    pressed.</p>
</body>

</html>
```

Event Object: Which element was clicked?

```
<html>
<head>
<script type="text/javascript">
function whichElement(e) {
    var targ
    if (!e) var e = window.event
    if (e.target) targ = e.target
        else if (e.srcElement) targ = e.srcElement
    if (targ.nodeType == 3) // defeat Safari bug
        targ = targ.parentNode
    var tname
   tname=targ.tagName
   alert("You clicked on a " + tname + " element.")
}
</script>
</head>

<body onmousedown="whichElement(event)">
<p>Click somewhere in the document. An alert box will alert the tag name of the element you clicked on.</p>

<h3>This is a header</h3>
<p>This is a paragraph</p>
<img border="0" src="ball16.gif" width="29" height="28" alt="Ball">
</body>
</html>
```

Event Object: Which event type occurred?

```
<html>
<head>

<script type="text/javascript">
    function whichType(event) {
        alert(event.type)
    }
</script>
</head>

<body onmousedown="whichType(event)">

<p>
Click on the document. An alert box will alert which type of event occurred.
</p>

</body>
</html>
```

Lab:

Exercise 6: Event object
4264_javascript_dom.zip

Code with Passion!
JPassion.com