JavaScript and HTML Documents.

Document Object Model:

DOM is an application Programming Interface that defines an interface b/w HTML document and Application Programs.

Specifications (versions of document object model).

Versions:

DOM0 → Supports IE 3.0, Netscape 3.0 and partially implemented in HTML 4 specification. HTML & JavaScript

DOM1 : developed in oct 1998 and specifications are HTML & XML.

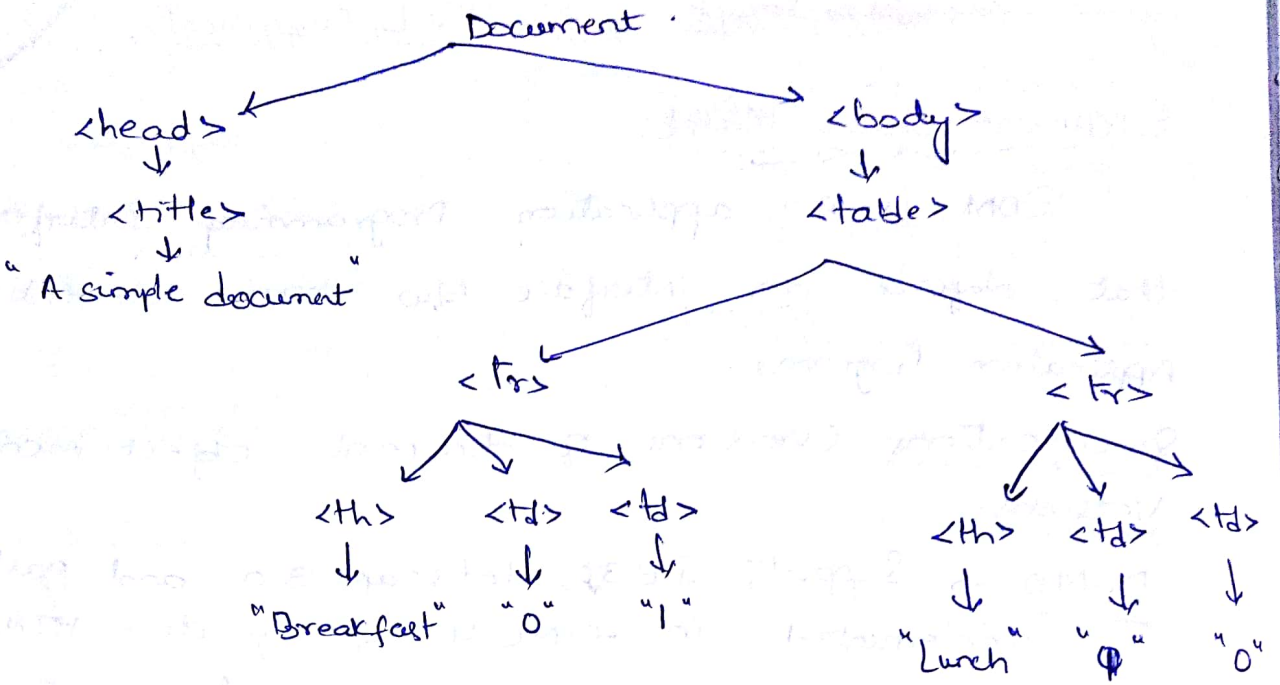
DOM2 : developed in NOV 2000, used in firefox with Style Sheet object model.

DOM3 : It is used for context model for XML documents and its still under development.

* Users can change, delete, add elements to create documents in an easier way.
* Documents in the DOM have a tree like structure, there can be more than one tree in a document.
* Dom is an abstract interface, it does not dictate that documents must be implemented as trees or collections of trees.

Eg: HTML document and DOM tree :

```
<html> <head> <title> A simple document </title>
</head> <body> <table>
<tr> <th> Breakfast </th> <td> 0 </td> <td> 1 </td>
</tr>
<tr> <th> Lunch </th> <td> 1 </td> <td> 0 </td> </tr>
</table> </body> </html>.
```



The DOM structure for a suiple document.

* In the JS binding to the DOM, the elements of a document are objects, with both data and operation
* Data are called properties
* Operations are called methods.

Eg: The elements would be represented as an object with 2 properties type and name, with the values "text" and "address",

<input type = "text" name = "address">.

Element Access in JavaScript:

The object associated with and HTML form element can be addressed in JavaScript.

* The way is to use the forms & elements arrays of the Document object, which refered through the document property of window object.

Eg: <html> <head> <title> Access to form elements </title>
</head> <body>
<form action = " ">
<input type = "button" name = "turnItOn" />
</form> </body> </html>.

* The address of JS object, associated with HTML element as its DOM address.

* The DOM address of button, using forms & elements arrays as follows:

var dom = document.forms[0].elements[0];

Problem: Element address, DOM address can be changed.

Eg: If a new button added before the turnItOn button in the document, DOM address shown wrong.

Another approach: To DOM addressing is to use element names.

Eg: <html> <head> <title> Access to form elements <Title> </head>
<body> <form name = "myForm" action = " ">
<input type = "button" name = "turnItOn">
</form> </body> </Html>:

Using name attributes, the button's DOM address:

var dom = document.myForm.turnItOn;

Drawback: The HTML 1.1 standard does not allow the name attribute in the form element. This is a validation problem, but causes no difficulty for browsers.

Another Approach: Element addressing is to use the JS method getElementById, defined in DOM 1.

Eg: If the id attribute of button set to "turnItOn", the DOM address of that button element:

```
var dom = document.getElementById("turnItOn");
```

* The parameter of getElementById can be any Expression that evaluates to a String, else it is a variable.
* ids are most useful for DOM addressing and names are required for form processing code; form elements often have both ids and names, both set to same value.
* Alternative to names & ids is provided by implicit arrays associated with checkbox & radio button group.
* To access the arrays, the DOM address of form object first must be obtained.

Eg:
```
<form id = "vehicleGroup">
    <input type = "checkbox" name = "vehicles"
        value = "Car" /> Car
<input type = "checkbox" name = "vehicles" value = "Truck" />
    Truck.
<input type = "checkbox" name = "vehicles" value = "Bike" />
    Bike.
</form>.
```

The implicit array, vehicles;
has 3 elements, which reference the 3 objects associated with 3 checkbox elements in group.

* The checked property of checkbox object is set to true if the button is checked.

Eg: var numChecked = 0;
var dom = document.getElementById("vehicleGroup");
for (index = 0; index < dom.vehicles.length; index++)
if (dom.vehicles[index].checked)
numchecked++;

Events and Event Handling:

Event: An event is an object that is implicitly created by the browser and the JavaScript system in response to something.

Eg: completion of loading of a document or a browser user action, such as mouse click on form button.

Event Handler: It is a script that is implicitly executed in response to the appearance of an event.

* Event Handler enables a Web document to be responsive to browser & user activities.

Use: To check for simple errors and omissions in user input to the elements of a form, when changed or when a form is submitted.

Events, Attributes and Tags:

A collection of events, which browsers implement and with which JS deals.

* These events associated with HTML tag attributes, used to connect the events to handlers.

* These appear in several diff. tags; and an event is created are related to a tag & an attribute.

* HTML text element is said to get focus when user puts mouse cursor over it & clicks the left mouse button.

Table 5.1 Events and their tag attributes

Event	Tag Attribute
blur	onblur
change	onchange
click	onclick
dblclick	ondblclick
focus	onfocus
keydown	onkeydown
keypress	onkeypress
keyup	onkeyup
load	onload
mousedown	onmousedown
mousemove	onmousemove
mouseout	onmouseout
mouseover	onmouseover
mouseup	onmouseup
reset	onreset
select	onselect
submit	onsubmit
unload	onunload

* The focus on an element can be forced with focus method.

* An element becomes blurred when the user moves the cursor away from the element and clicks the left mouse button.

* Hortext elements can also have focus, but the condition is less useful.

* These are Case sensitive

Eg: click ✓

Click ×.

Table 5.2 Event attributes and their tags

Attribute	Tag	Description
onblur	<a>	The link loses the input focus
	<button>	The button loses the input focus
	<input>	The input element loses the input focus
	<textarea>	The text area loses the input focus
	<select>	The selection element loses the input focus
onchange	<input>	The input element is changed and loses the input focus
	<textarea>	The text area is changed and loses the input focus
	<select>	The selection element is changed and loses the input focus
onclick	<a>	The user clicks on the link
	<input>	The input element is clicked
ondblclick	Most elements	The user double clicks the left mouse button
onfocus	<a>	The link acquires the input focus
	<input>	The input element receives the input focus
	<textarea>	A text area receives the input focus
	<select>	A selection element receives the input focus
onkeydown	<body>, form elements	A key is pressed down
onkeypress	<body>, form elements	A key is pressed down and released
onkeyup	<body>, form elements	A key is released
onload	<body>	The document is finished loading

Table 5.2 Event attributes and their tags (*continued*)

Attribute	Tag	Description
onmousedown	Most elements	The user clicks the left mouse button
onmousemove	Most elements	The user moves the mouse cursor within the element
onmouseout	Most elements	The mouse cursor is moved away from being over the element
onmouseover	Most elements	The mouse cursor is moved over the element
onmouseup	Most elements	The left mouse button is unclicked
onreset	<form>	The reset button is clicked
onselect	<input>	The mouse cursor is moved over the element
	<textarea>	The text area is selected within the text area
onsubmit	<form>	The *Submit* button is pressed
onunload	<body>	The user exits the document

Two ways to register an event handler in DOMO event.

① Assigning the event handler script to an event tag attribute.

```
<input type="button" id="myButton"
    onclick="alert('You clicked my button!');" />
```

* The handler consists of more than a single

Cont.

* A function used, and string value of attribute is the call to the fn. as follows.

```
<input type="button" id="myButton"
                onclick="myButtonHandler();" />
```

* Event handler is registered by assignment to event property on button object.

```
document.getElementById("myButton").onclick =
                                    myButtonHandler;
```

* The stmt must follow both
  ① the handler function
  ② the form element, so JS has seen both before assigning the property.
* Name of the handler fn is assigned to the property: string or call to fn.

Handling Events from Body Elements:

① load & ② unload.

Eg: an alert message when the body of document has been loaded.

So, onload attribute of <body> to specify the event handler.

* The onload event is useful than the load event.

* Used to cleanup before a document is unloaded.

Eg: If the document opened a second browser window, that window should be an onload event handler.

```
<?xml version = "1.0"  encoding = "utf-8" ?>
<!DOCTYPE html PUBLIC "-//w3c//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- load.html
     A document for load.js
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <title> load.html </title>
    <script type = "text/javascript"  src = "load.js" >
    </script>
  </head>
  <body onload="load_greeting();">
    <p />
  </body>
</html>
```

```
// load.js
//   An example to illustrate the load event

// The onload event handler
function load_greeting () {
  alert("You are visiting the home page of \n" +
        "Pete's Pickled Peppers \n" + "WELCOME!!!");
}
```

Figure 5.2 shows a browser display of `load.html`.

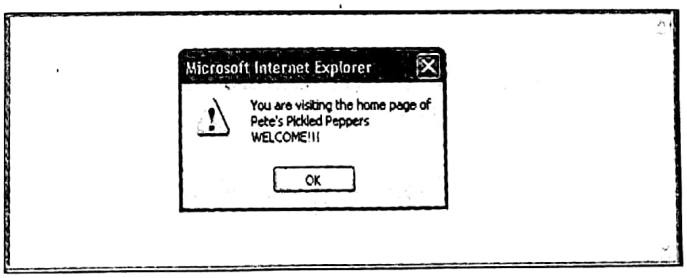
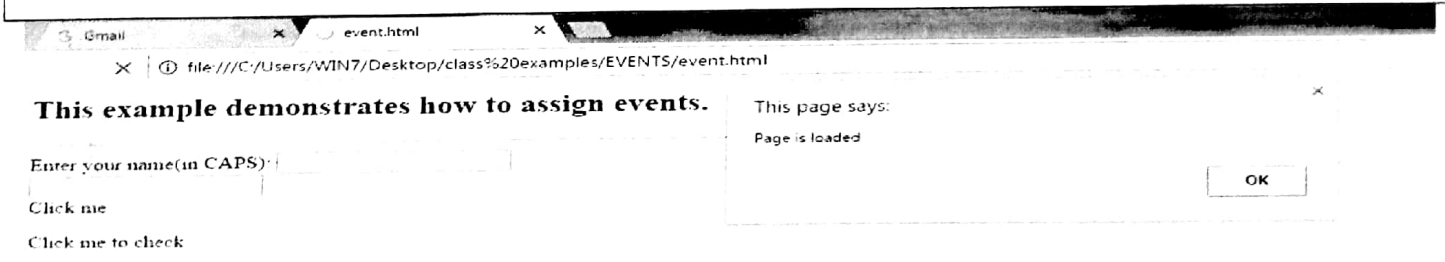
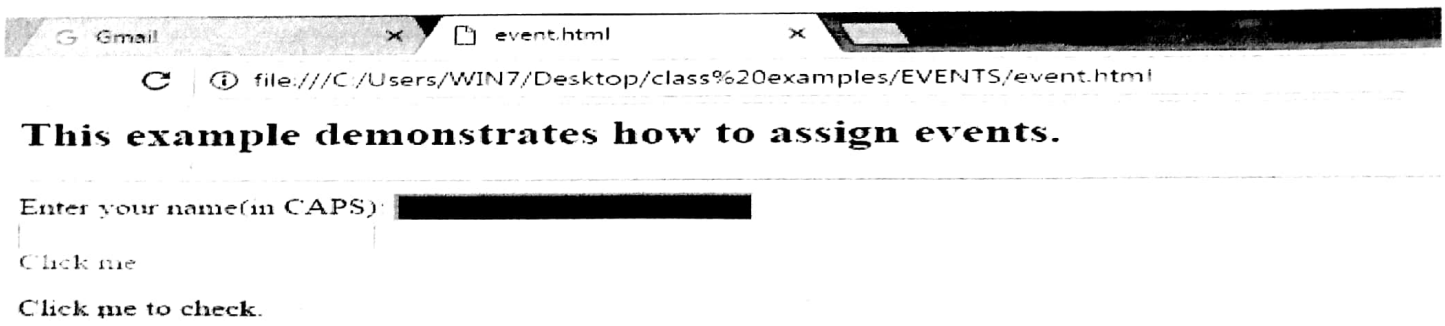


Figure 5.2 Display of `load.html`

Example : with onload, onfocus, onblur, onmousedown & onmouseup events.

```
<html>
<body onload="func1()">
<h2>This example demonstrates how to assign events.</h2>
<hr/>
Enter your name(in CAPS):
<input type="text" id="fname" onchange="func2()" onfocus="func3()"/><br>
<input type="text" id="fname" onblur="func4()"/><br>
<a id="second" onclick="func5()"/>Click me.</a><br>
<p id="third" onmousedown="func6()" onmouseup="func7()">Click me to check.</p>
<script>
function func1()
{  alert("Page is loaded");  }
function func2()
{  var x = document.getElementById("fname");
   x.value = x.value.toLowerCase();  }
function func3()
{  document.getElementById("fname").style.backgroundColor="red";  }
function func4()
{alert("Input field lost focus.");}
function func5()
{ document.getElementById("second").style.color = "red";  }
function func6()
{  document.getElementById("third").style.backgroundColor = "blue";  }
function func7()
{  document.getElementById("third").style.backgroundColor = "yellow";  }
</script>
</body>
</html>
```





Handling Events from Button Elements:

Buttons in Web document provide simple & effective way to collect simple input from the browser user.

* Used event is created by button actions is click.
* The below eg. shows a plain button.
* A set of radio buttons enables user to choose info. about specific airplane.
* click event is used to trigger a call to alert, gives description of selected airplane.
* The event handlers send the value of the pressed radio button to handler.
* In radio_click.html the event handler is registered by assigning its call to onclick attribut of radio button.
* If button is clicked, identified by parameter sent in handler call in button element.
* Alternative, to include code in handler to determine which radio button was pressed.

```
<?xml version = "1.0" encoding = "utf-8" ?>
<!DOCTYPE html PUBLIC "-//w3c//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- radio_click.hmtl
     A document for radio_click.js
     Creates four radio buttons that call the planeChoice
     event handler to display descriptions
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <title> radio_click.html </title>
  <script type = "text/javascript"  src = "radio_click.js" >
  </script>
  </head>
  <body>
    <h4> Cessna single-engine airplane descriptions </h4>
    <form id = "myForm"  action = "">
      <p>
        <label> <input type = "radio"  name = "planeButton"
                       value = "152"
                       onclick = "planeChoice(152)" />
        Model 152 </label>
        <br />
        <label> <input type = "radio"  name = "planeButton"
                       value = "172"
                       onclick = "planeChoice(172)" />

        Model 172 (Skyhawk) </label>
        <br />
        <label> <input type = "radio"  name = "planeButton"
                       value = "182"
                       onclick = "planeChoice(182)" />

        Model 182 (Skylane) </label>
        <br />
        <label> <input type = "radio"  name = "planeButton"
                       value = "210"
                       onclick = "planeChoice(210)" />

        Model 210 (Centurian) </label>
      </p>
    </form>
  </body>
</html>
```

* A browser display of radio_click.html, shows the alert window that results from choosing the Model 182 radio button in radio_click.html.

```
// radio_click.js
//   An example of the use of the click event with radio buttons,
//   registering the event handler by assignment to the button
//   attributes

// The event handler for a radio button collection
function planeChoice (plane) {

// Produce an alert message about the chosen airplane
  switch (plane) {
    case 152:
      alert("A small two-place airplane for flight training");
      break;
    case 172:
      alert("The smaller of two four-place airplanes");
      break;
    case 182:
      alert("The larger of two four-place airplanes");
      break;
    case 210:
      alert("A six-place high-performance airplane");
      break;
    default:
      alert("Error in JavaScript function planeChoice");
      break;
  }
}
```

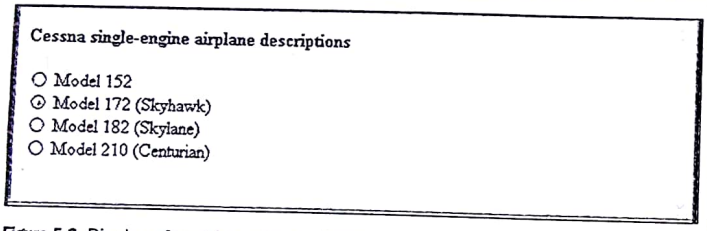


Figure 5.3 Display of radio_click.html

Figure 5.4 The result of pressing the Model 182 button in radio_click

→ Next example: radio_click2.html, is same as radio_click.html, registers the event handler by assigning the name of the handler to the event properties of radio button objects.

Eg: Registering the handler on first radio button

```
document.getElementById("myform").elements[0].onclick =
                                        planeChoice;
```

* The Eg uses 3 files,

① for HTML
② for script for event handler
③ for script to register the handlers.

```
<!-- radio_click2.hmtl
     A document for radio_click2.js
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <title> radio_click2.html </title>

<!-- Script for the event handler -->
  <script type = "text/javascript"  src = "radio_click2.js" >
  </script>

  </head>
  <body>
    <h4> Cessna single-engine airplane descriptions </h4>
    <form id = "myForm"  action = "">
      <p>
        <label> <input type = "radio"  name = "planeButton"
                       value = "152" />
        Model 152 </label>
        <br />
        <label> <input type = "radio"  name = "planeButton"
                       value = "172" />
        Model 172 (Skyhawk) </label>
        <br />
        <label> <input type = "radio"  name = "planeButton"
                       value = "182" />
        Model 182 (Skylane) </label>
        <br />
        <label> <input type = "radio"  name = "planeButton"
                       value = "210" />
        Model 210 (Centurian) </label>
      </p>
    </form>

<!-- Script for registering the event handlers -->
    <script type = "text/javascript"  src = "radio_click2r.js" >
    </script>

  </body>
</html>
```

* In radio_click2.js the handler includes a loop to determine which radio button created the click event.

```
// radio_click2.js
//   An example of the use of the click event with radio buttons,
//   registering the event handler by assigning an event property

// The event handler for a radio button collection
function planeChoice (plane) {

// Put the DOM address of the elements array in a local variable
  var dom = document.getElementById("myForm");

// Determine which button was pressed
  for (var index = 0; index < dom.planeButton.length;
       index++) {
    if (dom.planeButton[index].checked) {
      plane = dom.planeButton[index].value;
      break;
    }
  }

// Produce an alert message about the chosen airplane
  switch (plane) {
    case "152":
      alert("A small two-place airplane for flight training");
      break;
    case "172":
      alert("The smaller of two four-place airplanes");
      break;
    case "182":
      alert("The larger of two four-place airplanes");
      break;
    case "210":
      alert("A six-place high-performance airplane");
      break;
    default:
      alert("Error in JavaScript function planeChoice");
      break;
  }
}
```

```
// radio_click2r.js
//   The event registering code for radio_click2
var dom = document.getElementById("myForm");
dom.elements[0].onclick = planeChoice;
dom.elements[1].onclick = planeChoice;
dom.elements[2].onclick = planeChoice;
dom.elements[3].onclick = planeChoice;
```

In radio_click2r.js, the form elements (radio buttons) are addressed as elements of the elements array.

* Other way, to give each radio button an id attribute & register the handler using id.

Eg: first radio button could be defined as:

<input type = "radio" name = "planeButton" value = "152" id = "152" />.

→ Then the event handler registration:

```
var dom = document.getElementById("myForm");
dom.getElementById("152").onclick = planeChoice;
dom.getElementById("172").onclick = planeChoice;
dom.getElementById("182").onclick = planeChoice;
dom.getElementById("210").onclick = planeChoice;
```

* There is no way to specify parameters on handler fn, when it is registered by assigning its name to event property.

Disadvantage: Event handler ~~of this~~ that are registered this way cannot use parameters.

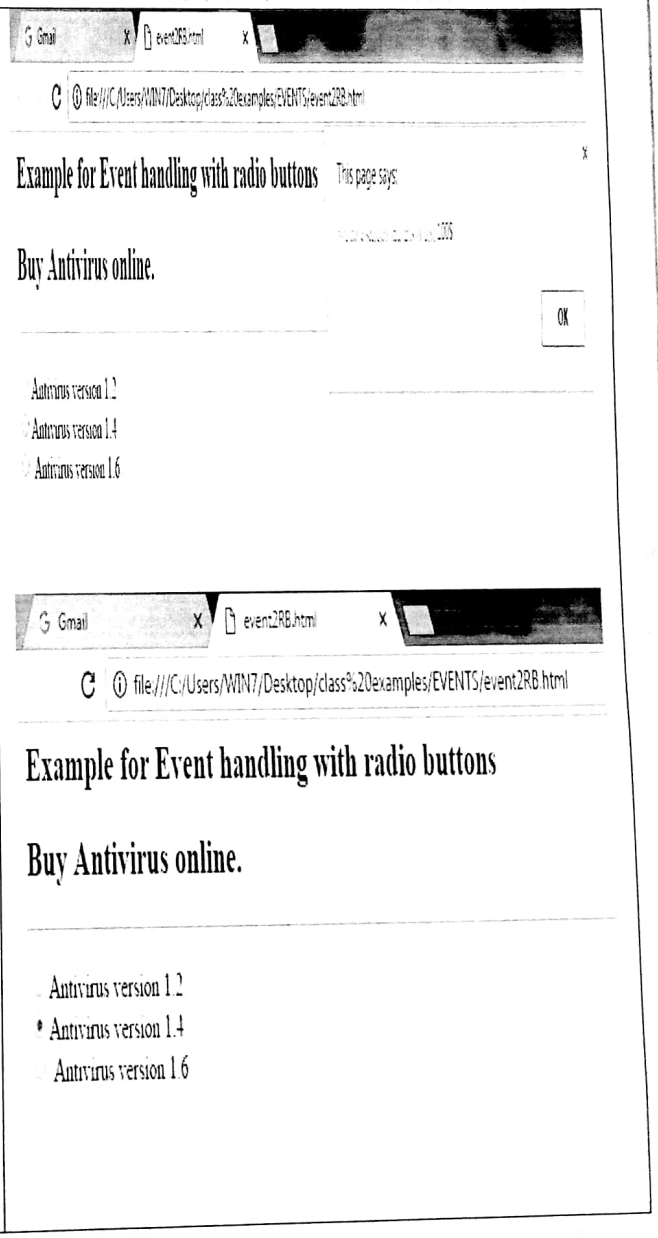
Two advantages: To registering handlers as properties as registering in HTML attributes.

① HTML & JavaScript seperated in the document which allows a kind of modularization of HTML documents, resulting in a cleaner design.

② Having handler fn. registered as the value of a property allows for the possibility of changing it during use.

→ Other Example :

```
<html>
<head>
<h2>Example for Event handling with radio
buttons</h2>
<script>
function choice(cost)
{  switch(cost)
   {   case 50:alert("you have selected antivirus
of cost  50$");break;
case 100:alert("you have selected antivirus of
cost     100$");break;
case 150:alert("you have selected antivirus of
cost     150$");break;
default:alert("error---");break;        }  }
</script>
</head>

<body>
<h2>Buy Antivirus online.</h2>
<hr/>
<form>
<p>
<input type="radio" name="antivirus" value="50"
onclick="choice(50)"/><label>Antivirus version
1.2</label><br />
<input type="radio" name="antivirus"
value="100"
onclick="choice(100)"/><label>Antivirus version
1.4</label><br />
<input type="radio" name="antivirus"
value="150"  onclick="choice(150)"/>
<label>Antivirus version 1.6</label></p>
</form>
</body>
</html>
```



Handling Events from Text Box and Password Elements:

Text boxes and passwords creat 4 diff events.

① blur, ② focus ③ change and ④ select.

The focus Event:

To do total cost and display to customer before order submitted to server for processing, Javascript is used.

* So that no other user can change the data.

* The change to a text box can be prevented by an event handler that blurs the text box every time the user attempts to put it in focus.

* Blur can be on a element using blur method.

```
<?xml version = "1.0"  encoding = "utf-8" ?>
<!DOCTYPE html PUBLIC "-//w3c//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- nochange.html
     A document for nochange.js
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head> <title> nochange.html </title>

<!-- Script for the event handlers -->
    <script type = "text/javascript"  src = "nochange.js" >
    </script>

 </head>
 <body>
   <form action = "">
     <h3> Coffee Order Form </h3>

<!-- A bordered table for item orders -->
     <table border = "border">

<!-- First, the column headings -->
       <tr>
```

```
          <th> Product Name </th>
          <th> Price </th>
          <th> Quantity </th>
        </tr>

<!-- Now, the table data entries -->
        <tr>
          <th> French Vanilla (1 lb.) </th>
          <td> $3.49 </td>
          <td> <input type = "text"  id = "french"
                      size ="2" /> </td>
        </tr>
        <tr>
          <th> Hazlenut Cream (1 lb.) </th>
          <td> $3.95 </td>
          <td> <input type = "text"  id = "hazlenut"
                size = "2" /> </td>
        </tr>
        <tr>
          <th> Colombian (1 lb.) </th>
          <td> $4.59 </td>
          <td> <input type = "text"  id = "colombian"
                size = "2" /></td>
        </tr>
      </table>

 <!-- Button for precomputation of the total cost -->
      <p>
        <input type = "button"  value = "Total Cost"
               onclick = "computeCost();" />
        <input type = "text"  size = "5"  id = "cost"
               onfocus = "this.blur();" />
      </p>

 <!-- The submit and reset buttons -->
      <p>
        <input type = "submit"  value = "Submit Order" />
        <input type = "reset"  value = "Clear Order Form" />
      </p>
    </form>
  </body>
</html>
```

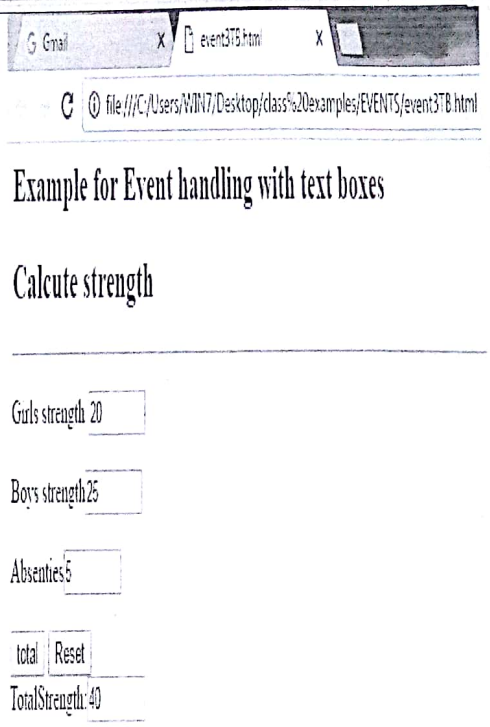
This example, the button labeled Total cost allows the user to precompute the total cost of the order.

```
// nochange.js
//   This script illustrates using the focus event
//   to prevent the user from changing a text field

// The event handler function to compute the cost
function computeCost() {
  var french = document.getElementById("french").value;
  var hazlenut = document.getElementById("hazlenut").value;
  var columbian = document.getElementById("colombian").value;

// Compute the cost
  document.getElementById("cost").value =
  totalCost = french * 3.49 + hazlenut * 3.95 +
              colombian * 4.59;
} //* end of computeCost
```

* The text box acquires focus, it is forced to blur with the blur method, prevents the user from changing the value.

Other Example:

```
<html><head>
<h2>Example for Event handling with text boxes</h2>
<script>
function totalstr()
{
varboys=parseInt(document.getElementById("boys").value)
;
var girls=parseInt(document.getElementById("girls").value);
var absc=document.getElementById("absc").value;
var res=boys+girls-absc;
document.getElementById("tot").value = res;      }
</script></head>
<body>
<h2>Calcute strength</h2>
<hr/>
<form id="myform">
<p>Girls strength
<input type="text" id="girls" size="5" ></p>
<p>Boys strength
<input type="text" id="boys" size="5" ></p>
<p>Absenties<input type="text" id="absc" size="5" ></p>
<p>
<input type="button" value="total"onclick="totalstr();">
<input type="reset"><br/>
TotalStrength:<input type="text" size="5" id="tot">
</p>
</form>
</body>
</html>
```



Validating from Input :

Validating form data on the client results in quicker response to users.

* when a user fills in a form input element incorrectly, a JS event-handler fn detects the error, the fn should do several things.

① It should produce an alert message indicating the error to the user and specify the correct format of input

② It should cause the input element to be put in focus, position the cursor in the element.

* This is done by focus method, called from DOM address of element.

Eg: If the element's id is phone, the element can put in focus:

document.getElementById("phone").focus();

(this puts the cursor in the phone text box).

③ The fn. should select the element, highlights the text in element. Done by select method.

document.getElementById("phone").select();

* The event handler check the data always return false if they detect an error, and true if no error.

Example: The HTML document creates the text boxes for passwords, and Reset & submit buttons & 2 scripts for the event handler for pswd_check.html and the event handler registrations.

```
<?xml version = "1.0" encoding = "utf-8" ?>
<!DOCTYPE html PUBLIC "-//w3c//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- pswd_chk.html
     A document for pswd_chk.ps
     Creates two text boxes for passwords
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <title> Illustrate password checking> </title>
    <script type = "text/javascript"  src = "pswd_chk.js" >
```

```
    </script>
  </head>
  <body>
    <h3> Password Input </h3>
    <form id = "myForm"  action = "" >
      <p>

      <label> Your password
        <input type = "password" id = "initial"
               size = "10" />
      </label>
      <br /><br />

      <label> Verify password
        <input type = "password"  id = "second"
               size = "10" />
      </label>
      <br /><br />

      <input type = "reset"  name = "reset" />
      <input type = "submit"  name = "submit" />
      </p>
    </form>

<!-- Script for registering the event handlers  -->
    <script type = "text/javascript"  src = "pswd_chkr.js">
    </script>

  </body>
</html>
```

```
// pswd_chk.js
//   An example of input password checking, using the submit
//   event

// The event handler function for password checking
function chkPasswords() {
  var init = document.getElementById("initial");
  var sec = document.getElementById("second");
  if (init.value == "") {
    alert("You did not enter a password \n" +
          "Please enter one now");
    init.focus();
    return false;
  }
  if (init.value != sec.value) {
    alert("The two passwords you entered are not the same \n" +
          "Please re-enter both now");
    init.focus();
    init.select();
    return false;
  } else
    return true;
}
```

```
// pswd_chkr.js
//   Register the event handlers for pswd_chk.html

document.getElementById("second").onblur = chkPasswords;
document.getElementById("myForm").onsubmit = chkPasswords;
```

5.6 shows browser display result from pressing the Query button on pswd_chk.html after diff pswds have been entered.

Figure 5.5 shows a browser display of pswd_chk.html after the two word elements have been input but before *Submit Query* has been clicked.

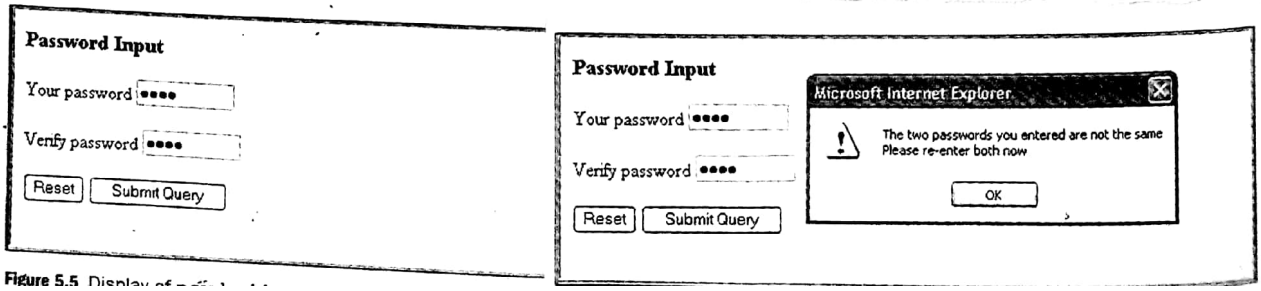


Figure 5.5 Display of

An example, checks the validity of form values for a name and phone number from text widgets.

* This used to check the forms input when values of text boxes are changed, detected by appearence of change event.

* If an error detected, an about message is generated & both focus and select are called to prompt.

* 3 fields created for correct format for the name is last-name, first-name, middle-initial.

* The pattern for matching such names is:

/^[A-Z][a-z]+,?[A-Z][a-z]+,?[A-Z]\.?$/

* Use of The anchors, ^ & $ on the ends of the pattern

* ? → for after spaces and after the period, means zero or one of qualified subpattern

* / → the period is backslashed so it matches only a period.

* Correct format of phone number is 3 digits & a dash, 3 digit and a dash and 4 digits.

* No characters are precede or follow phone number.

* Pattern: /^d{3}-\d{3}-\d{4}$/

A HTML document, validator.html, displays the text boxes for a customer's name & phone no.

```
<?xml version = "1.0" encoding = "utf-8" ?>
<!DOCTYPE html PUBLIC "-//w3c//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- validator.html
     A document for validator.js
     Creates text boxes for a name and a phone number
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <title> Illustrate form input validation> </title>
    <script type = "text/javascript"  src = "validator.js" >
    </script>
  </head>
  <body>
    <h3> Customer Information </h3>
    <form action = "">
      <p>
        <label>
          <input type = "text"  id = "custName" />
          Name (last name, first name, middle initial)
        </label>
        <br /><br />

        <label>
          <input type = "text"  id = "phone" />
          Phone number (ddd-ddd-dddd)
        </label>
        <br /><br />

        <input type = "reset"  id = "reset" />

        <input type = "submit"  id = "submit" />
      </p>
    </form>

  <!-- An inline script for the event handler registrations -->
    <script type = "text/javascript">
      <!--
  // Set form element object properties to their
  // corresponding event handler functions

        document.getElementById("custName").onchange = chkName;

      document.getElementById("phone").onchange = chkPhone;
    // -->
    </script>
  </body>
</html>
```

The following is the script for the event handlers for `validator.html`.

```
// validator.js
//   An example of input validation using the change and submit
//   events

// The event handler function for the name text box
function chkName() {
  var myName = document.getElementById("custName");

// Test the format of the input name
//  Allow the spaces after the commas to be optional
//  Allow the period after the initial to be optional
  var pos = myName.value.search(
            /^[A-Z][a-z]+, ?[A-Z][a-z]+, ?[A-Z]\.?$/);
  if (pos != 0) {
    alert("The name you entered (" + myName.value +
          ") is not in the correct form. \n" +
          "The correct form is: " +
          "last-name, first-name, middle-initial \n" +
          "Please go back and fix your name");
    myName.focus();
    myName.select();
    return false;
  } else
    return true;
}

// The event handler function for the phone number text box
function chkPhone() {
  var myPhone = document.getElementById("phone");

// Test the format of the input phone number
  var pos = myPhone.value.search(/^\d{3}-\d{3}-\d{4}$/);
  if (pos != 0) {
    alert("The phone number you entered (" + myPhone.value +
          ") is not in the correct form. \n" +
          "The correct form is: ddd-ddd-dddd \n" +
          "Please go back and fix your phone number");

  myPhone.focus();
  myPhone.select();
  return false;
} else
  return true;
}
```

5.7 shows browser screen of validator.html

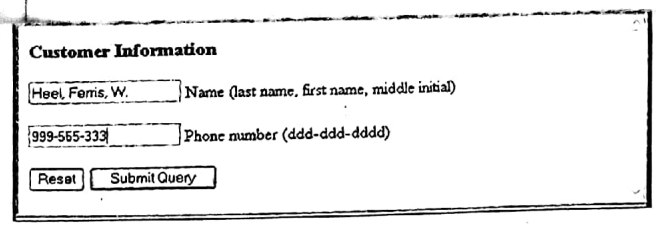


Figure 5.7 Display of `validator.html`, with an invalid phone number, while the phone text field has focus

Figure 5.8 shows the `alert` dialog box generated by pressing the Enter button in the phone text field of the screen of Figure 5.7.

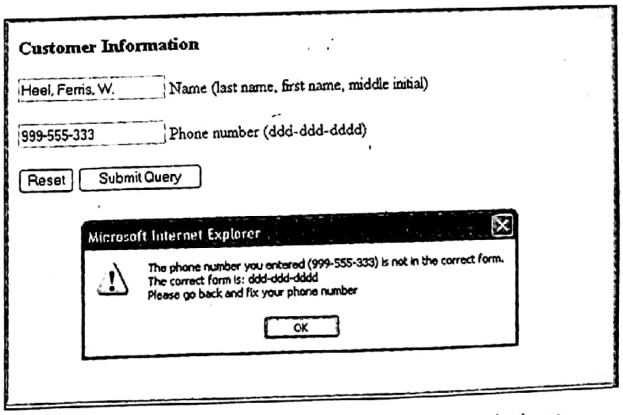


Figure 5.8 The message created by entering an invalid telephone number in `validator.html`

The Navigator Object :

navigator object indicates which browser is being used to view HTML document.

* Browser's name is stored in the appName property of navigator object.
* The version of browser is stored in appVersion property of navigator object.

Example: The use of navigator, to display the browsername & version number.

```
<?xml version = "1.0"  encoding = "utf-8" ?>
<!DOCTYPE html PUBLIC "-//w3c//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<!-- navigate.html
     A document for navigate.js
     Calls the event handler on load
     -->
<html xmlns = "http://www.w3.org/1999/xhtml">
  <head>
    <title> navigate.html </title>
    <script type = "text/javascript"  src = "navigate.js" >
    </script>
  </head>
  <body onload = "navProperties()">
  </body>
</html>
```

```
// navigate.js
//  An example of using the navigator object

// The event handler function to display the browser name
//  and its version number
function navProperties() {
  alert("The browser is: " + navigator.appName + "\n" +
    "The version number is: " + navigator.appVersion + "\n");
}
```

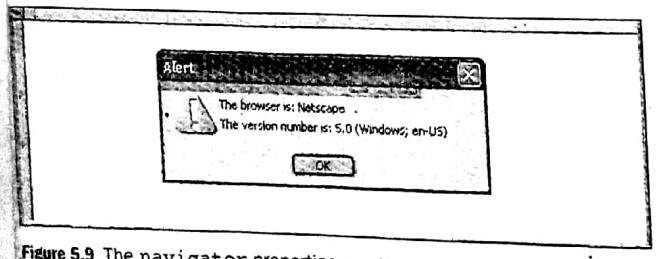


Figure 5.9 The navigator properties appName and appVersion for Firefox 2

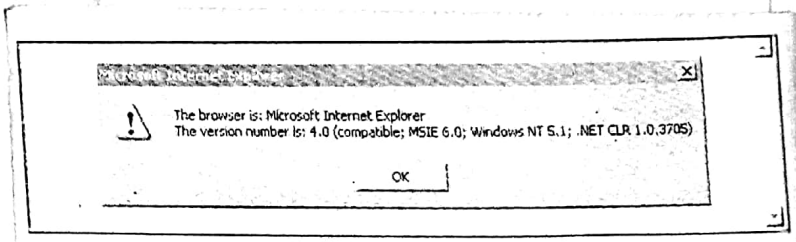
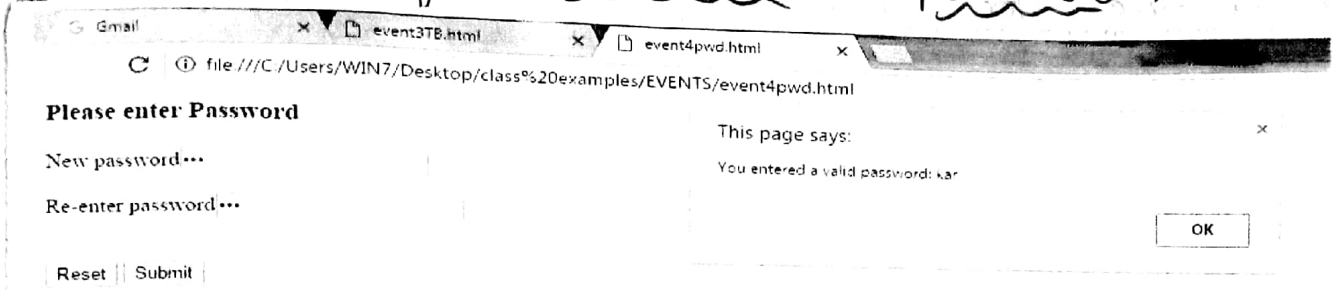


Figure 5.10 The navigator properties appName and appVersion for Internet Explorer 7

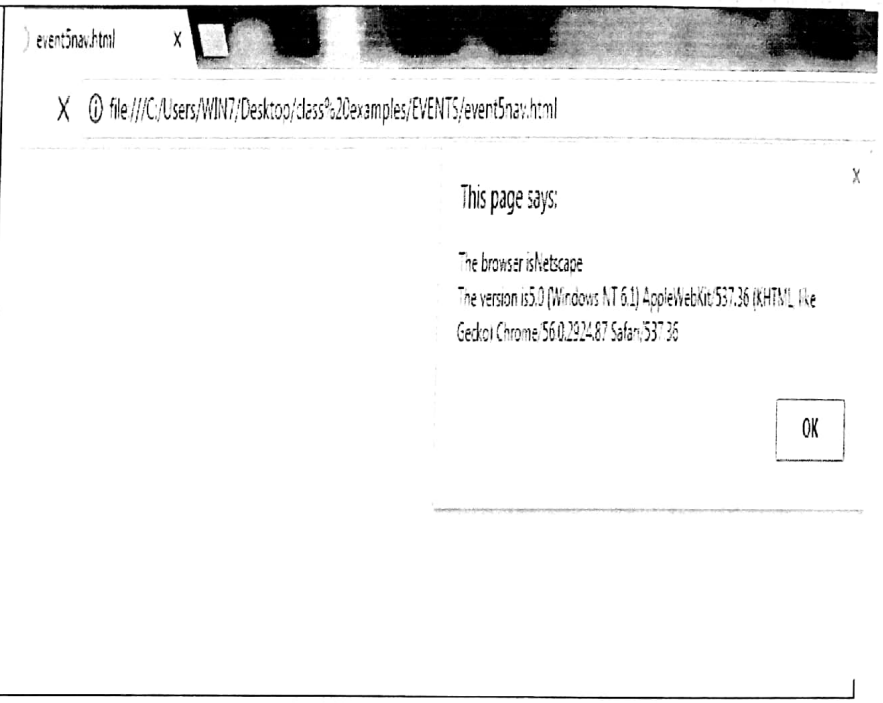
Other Example for Validation : Password.

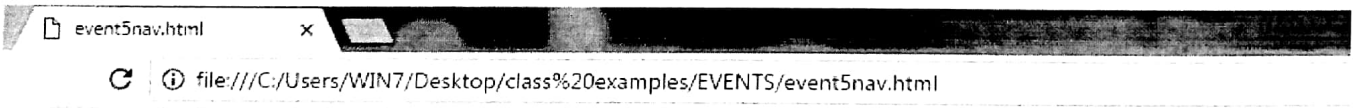



```
<html><head>
<script>
        function chkpwd(form)
{
  if(form.first.value=="")
{
alert("you did not enter password \n" + "Pls Enter it");
form.first.focus();
return false;
   }
 if(form.first.value!=form.second.value)
{.
   alert("passwords not match \n" + "Pls Re-Enter it");
form.first.focus();
return false;
   }
else
{
 alert("You entered a valid password: " + form.first.value);
return true;
  }
}
</script></head>
<body>
<h3>Please enter Password</h3>
<form onsubmit="return chkpwd(this);">
<p><label>New password</label>
<input type="password" name="first"></p>
<p><label>Re-enter password</label>
<input type="password" name="second"></p>
<br/>
<input type="reset" name="reset"/>
<input type="submit" name="submit"/>
</form>
</body>
</html>
```

Other example for navigator object :

```
<html>
<body onload="navP()">
<h2>Example for navigator object to
determine the browser & its
version</h2>
</body>
<script>
function navP()
{
alert("The browser is"+
navigator.appName+"\n"+"The
version is"+navigator.appVersion);
}
document.write("The browser is"+
navigator.appName+ "<br/>" +"The
version is"+navigator.appVersion);
</script>
</html>
```





Example for navigator object to determine the browser & its version

The browser isNetscape
The version is5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36