Web Programming

Lecture 5 – Dynamic Documents with JavaScript

Dynamic HTML

- Dynamic HTML is a set of technologies that allows dynamic changes to HTML documents.
- An embedded script can be used to change tag attributes, contents or element style properties.
- These changes are not uniformly supported across the full spectrum of browsers.
  - Most modern browsers support DOM 0 model.
  - DOM 2 is supported by Firefox 2 but not Internet Explorer 8.

What can you do with Dynamic HTML?

- Elements can be moved to new positions on the display.
- Elements can be made to appear and disappear.
- Foreground (text) and background colors can be changed.
- The font, font size and font style an be changed.
- Element content can be changed.
- The stacking order of overlapping elements (such as pictures) can be changed.
- The mouse cursor position can be changed.
- Text can be made to move across the display.

Positioning Elements

- Before HTML 4.0, the only control that existed for positioning document elements were tables (slow to load) and frames (deprecated).
- CSS-P (***C***ascading ***S***tyle ***S***heets **–** ***P***ositioning) are fully supported by IE 7 and Firefox 2, and provide style properties that allow for positioning and repositioning of document elements:
  - **`left`** - the distance from the top of some reference point
  - **`top`** - the distance from the top of some reference point
  - **`position`** – can be **`absolute`**, **`relative`**, or **`static`**.

Absolute Positioning

- Absolute positioning allows elements to be position on a display without regard to other elements of the document.
- It can be used to create a "watermark" within a document.
- The **`width`** property (which is set in the next example) limits how wide (on the screen) the element will be (instead of using automatic wordwrap).
- When an element is position absolutely inside another ***positioned*** element, the top and left properties are relative to the top left corner of the element in which it is contained.

absPos.html

```
<!DOCTYPE html>

<!-- absPos.html
    Illustrates absolute positioning  of elements
  -->
<html lang = "en">
  <head>
    <title> Absolute positioning </title>
    <meta charset = "utf-8">
    <style type = "text/css">
      /* A style for a paragraph of text */
      .regtext {font-family: Times;
                font-size: 1.2em; width: 500px}
      /* A style for the text to be absolutely
        positioned */
      .abstext {position: absolute; top: 25px;
                left: 25px;
                font-family: Times;
                font-size: 1.9em;
                font-style: italic;
                font-weight: bold;
                letter-spacing: 1em;
                color: rgb(160, 160, 160);
                width: 450px}
    </style>
  </head>
```

```
<body>
  <p class = "regtext">
    Apple is the common name for any tree of the
    genus Malus, of the family Rosaceae.  Apple
    trees grow in any of the temperate area of the
    world.  Some apple blossoms are white, but
    most have stripes or tints of rose.  Some
    apple blossoms are bright red.  Apples have a
    firm and fleshy structure that rows from the
    blossom.  The colors of apples range from
    green to very dark red.  The wood of apples
    trees is fine-grained and hard.  It is,
    therefore, good for furniture construction.
    Apple trees have been grown for many
    centuries.  They are propagated by grafting
    because they do not reproduce themselves.
  </p>
```

```
    <p class = "abstext">
      APPLES ARE GOOD FOR YOU
    </p>
  </body>
</html>
```

absPos2.html

```
<!DOCTYPE html>

<!-- absPos2.html
     Illustrates absolute positioning  of elements
     -->

<html lang = "en">
  <head>
    <title> Absolute positioning </title>
    <meta charset = "utf-8">
    <style type = "text/css">

      /* A style for a paragraph of text */
      .regtext {font-family: Times;
                font-size: 1.2em; width: 500px;
                position: absolute; top: 100px;
                left: 100px;}
      /* A style for the text to be absolutely
                     positioned */
      .abstext {position: absolute; top: 25px;
                left: 25px; font-family: Times;
                font-size: 1.9em;
                font-style: italic;
                letter-spacing: 1em;
                color: rgb(160, 160, 160);
                width: 450px}
    </style>
  </head>
```

```
<body>
  <div class = "regtext">
    Apple is the common name for any tree of the
    genus Malus, of the family Rosaceae.  Apple
    trees grow in anyof the temperate area of the
    world.  Some apple blossoms are white, but
    most have stripes or tints of rose.  Some
    apple blossoms are bright red.  Apples have a
    firm and fleshy structure that rows from the
    blossom.  The colors of apples range from
    green to very dark red.  The wood of apples
    trees is fine-grained and hard. It is,
    therefore, good for furniture construction.
    Apple trees have been grown for many
    centuries.  They are propagated by grafting
    because they do not reproduce themselves.
```

```
      <span class = "abstext">
        APPLES ARE GOOD FOR YOU
      </span>
    </div>
  </body>
</html>
```

Relative Positioning

- If an element's position is relative but **`top`** and **`left`** are not specified, it does not affect its position, but allows for it to be moved later.
- If top and left are specified, they indicated how far it is moved from where it would have been positioned.
- Negative values reposition the element up and to the left.

Uses for Relative Positioning

- Relative positioning can be used to create subscripts and superscripts (together with the **`<span>`** tag)

relPos.html

```
<!DOCTYPE html>

<!-- relPos.html
     Illustrates relative positioning  of elements
     -->
<html lang = "en">
  <head>
    <title> Relative positioning </title>
    <meta charset = "utf-8" />
    <style type = "text/css">
      .regtext {font: 2em Times}
      .spectext{font: 2em Times; color: red;
                position: relative; top: 15px;}
    </style>
  </head>
  <body>
    <p class = "regtext">
      Apples are <span class = "spectext">
          GOOD
      </span>  for you.
    </p>
  </body>
</html>;">
```

Static Positioning

- Static positioning is the default.
- A statically positioned tag is placed as if it had the position value of relative; however, it cannot have **`top`** and **`left`** properties initially set or changed.

Moving Elements

- An element with absolute or relative positioning can be moved by changing its values of **`top`** and/or **`left`**.
- With ***absolute*** positioning, it moves to the position indicate by the new values.
- With ***relative*** positioning, it moved by the amount indicated by the new values.

`mover.html`

```
<!DOCTYPE html>
<!-- mover.html
     Uses mover.js to move an images within a
  document
     -->
<html lang = "en">
  <head>
    <title> Moving elements </title>
    <meta charset = "utf-8" />
    <script type = "text/javascript"
            src = "mover.js">
    </script>
  </head>
```

```
 <body>
    <form action = "">
      <p>
        <label>
          x coordinates:
          <input type = "text" id = "leftCoord"
                 size = "3" />
        </label>
        <br />

        <label>
          y coordinates:
          <input type = "text" id = "topCoord"
                 size = "3" />
        </label>
        <br />
```

```
        <input type = "button" value = "Move it"
               onclick = "moveIt('nebula',
          document.getElementById('topCoord').value,
        document.getElementById('leftCoord').value)"
                 />
      </p>
    </form>

    <div id = "nebula" style = "position: absolute;
                       top: 115px; left: 0;">
      <img src = "ngc604.jpg"
           alt = "(Picture of a nebula)" />
    </div>
  </body>
</html>
```

mover.js

```
//  mover.js
//  Illustrates moving an element within a document

// The event handler function to move an element
function moveIt(movee, newTop, newLeft)  {
  dom = document.getElementById(movee).style;

//  change the top and left properties to perform
   the move
//  Note the addition of units to the input values
  dom.top = newTop + "px";
  dom.left= newLeft + "px";
}
```

Element Visibility

- Elements can be made invisible or visible by changing its style property **`visibility`** from **`visible`** to **`hidden`** or from **`hidden`** to **`visible`**.

showHide.html

```
<!DOCTYPE html>

<!-- showHide.html
     Uses showHide.js
     Illustrates visibility control of elements
     -->

<html lang = "en">
  <head>
    <title> Visibility control </title>
    <meta charset = "utf-8" />
    <script type = "text/javascript"
            src = "showHide.js">
    </script>
  </head>
```

```
  <body>
    <form action = "">
      <div id = "saturn" style = "position:relative;
                           visibility: visible;">
        <img src = "saturn.jpg"
             alt = "(Pictures of Saturn)" />
      </div>

      <p>
        <br />
        <input type = "button"
               value = "Toggle Saturn"
               onclick = "flipImag()" />
      </p>
    </form>
  </body>
</html>
```

showHide.js

```
//  showHide.js
//  Illustrates visibility coontrol of elements

//  The even handler function to toggle the
  visibility
//  of the images of Saturn
function flipImag()  {
  dom = document.getElementById("saturn").style;

  // Flip the visibility adjective to
  // whatever it is not now
  if (dom.visibility == "visible")
    dom.visibility = "hidden";
  else
    dom.visibility = "visible";
}
```

Changing Colors and Fonts

- Foreground (text) and background colors can both be changed.
- The various properties of a font can be changed including:
  - font (or font family
  - font size
  - font style (i. e., italic or normal)
  - font weight (i.e., bold or normal)

Changing Colors

- Background and foreground colors can be changed by assigning a new value to the properties **`backgroundColor`** or **`color`** respectively.
- These changes can be made using JavaScript
- The following example has the element call the function when a change is made to the entry in a text – **`this.value`** refers to the string in the current element

dynColors.html

```
<!DOCTYPE html>

<!-- dynColors.html
     Uses dynColors.js
     Illustrates dynamic foreground and background
  colors
     -->
<html lang = "en">
  <head>
    <title> Dynamic Colors </title>
    <meta charset = "utf-8">
    <script type = "text/javascript"
            src = "dynColors.js">
    </script>
  </head>
  <body>
    <p style = "font-family: Times;
                font-style: italic;
                font-size: 2em;">
      This small page illustrates dynamic setting of
      the foreground and background colors for a
      document
    </p>

    <form action = "">
      <p>
        <label>
          Background color:
          <input type = "text" name = "background"
                 size = "10"
                 onchange
                   = "setColor('background',
                               this.value)" />
        </label>
        <br />
```

```
        <label>
          Foreground color:
          <input type = "text" name = "foreground"
                 size = "10"
                 onchange
                      = "setColor('foreground',
                                  this.value)" />
        </label>
        <br />

      </p>
    </form>
  </body>
</html>
```

dynColors.js

```
// dynColor.js
// Illustrates dynamic foreground and background
   colors

// The event handler function to dynamically set the
// color of background or foreground
function setColor(where, newColor)  {
  if (where == "background")
    document.body.style.backgroundColor = newColor;
  else
    document.body.style.color = newColor;

}
```

Changing Fonts

- Any property of a link can be changed when the mouse is pointing to it by using the **`mouseover`** event to make a change.
- The properties can be changed back when the mouse is moved off the element by using the **`mouseout`** event.

dynFont.html

```
<!DOCTYPE html>
<!-- dynFont.html
     Illustrates dynamic font styles and colors
     -->

<html lang = "en">
  <head>
    <title> Dynamic fonts </title>
    <meta charset = "utf-8" />
    <style type = "text/css">
      .regText {font: 1.1em 'Times New Roman';}
    </style>
  </head>
```

```
  <body>
    <p class = "regText">
      The state of
      <a style = "color: blue;"
         onmouseover = "this.style.color = 'red';
                    this.style.fontStyle = 'italic'
                    this.style.fontSize = '2em';"
         onmouseout =  "this.style.color = 'blue';
                    this.style.fontStyle = 'normal'
                    this.style.fontSize = '1.1em';">
         Washington
      </a>
      produces many of our nation's apple.
    </p>
  </body>
</html>
```

Dynamic Content

- The content of an XHTML document element is accessed through the element's **`content`** property.
- Example
  - Assistance can be given to a user filling out a form by using a help box, whose contents will change depending on where the mouse is hovering.
  - This can be implemented using JavaScript functions called by the events **`mouseover`** and **`mouseout`**.

dynValue.html

```
<html lang = "en">
  <head>
    <title> Dynamic values </title>
    <meta charset = "utf-8" />
    <script type = "text/javascript" src =
  "dynValue.js">
    </script>
    <style type = "text/css">
      textarea {position: absolute; left: 250px;
                top: 0px;}
      span {font-style: italic;}
      p {font-weight: bold;}
    </style>
  </head>
```

```
<body>
  <form  action = "">
    <p>
      <span>
        Customer information
      </span>
      <br /> <br />

      <label>
        Name:
        <input type = "text"
               onmouseover = "messages(0)"
               onmouseout = "messages(4)" />
      </label>
      <br />
```

```
      <label>
        E-mail:
        <input type = "text"
               onmouseover = "messages(1)"
               onmouseout = "messages(4)" />
      </label>
      <br /> <br />

      <span>
        To create an account, provide the
        following:
      </span>
      <br /> <br />
```

```
        <label>
           User ID:
           <input type = "text"
                  onmouseover = "messages(2)"
                  onmouseout = "messages(4)" />
         </label>
         <br />

         <label>
           Password:
           <input type = "password"
                  onmouseover = "messages(3)"
                  onmouseout = "messages(4)" />
         </label>
         <br />
```

```
         <textarea id = "adviceBox" row = "3"
                   cols = "50">
           This box provides advice on filling out
           the form on this page.  Put the mouse
           cursor over any input field to get advice.
         </textarea>
         <br /> <br />
         <input type = "submit" value = "Submit" />
         <input type = "reset" value = "Reset" />
       </p>
     </form>
   </body>
</html>
```

dynValue.js

```
//  dynValue.js
//  Illustrates dynamic values

var helpers = ["Your name must be in the form: \n \
   first name, middle initial., last name",
   "Your email address must have the form: \
user@domain",
"Your user ID must have at least six \
characters and it must include one digit",
"Your password must have at least six \
characters and it must include one digit",
"This box provides advice on filling out\
the form on this page.  Put the mouse cursor over \
any input field to get advice"]
```

```
//**************************************************//
// The event handler function to change the value of
// the textarea

function messages(adviceNumber) {
  document.getElementById("adviceBox").value =
                     helpers[adviceNumber];
}
```

Stacking Elements

- While document elements are placed on the display in two dimensions, there is a third dimension that can be used in stacking one element over another.
- The order in which they are stacked is determined by a style property called z-index.
- The larger the z-index, the nearer the top of the stack the element will appear.

stacking.html

```
<!DOCTYPE html>

<!-- stacking.html
     Uses stacking.js
     Illustrates dynamic stackng of images
     -->
<html lang = "en">
  <head>
    <title> Dynamic stacking of images </title>
    <meta charset = "utf-8" />
    <script type = "text/javascript"
            src = "stacking.js">
    </script>
    <style type = "text/css">
      .plane1 {position: absolute; top: 0;
               left: 0; z-index: 0;}
      .plane2 {position: absolute; top: 50px;
               left: 110px; z-index: 0;}
      .plane3 {position: absolute; top: 100px;
               left: 220px; z-index: 0;}
    </style>
  </head>

  <body>
    <p>
      <img class = "plane1" id = "C172"
           src = "c172.jpg"
           alt = "(Picture of a C172)"
           onclick = "toTop('C172')" />
```

```
      <img class = "plane2" id = "Cix"
           src = "cix.jpg"
           alt = "(Picture of a Citation airplane)"
           onclick = "toTop('Cix')" />

      <img class = "plane3" id = "C182"
           src = "c182.jpg"
           alt = "(Picture of a C182)"
           onclick = "toTop('C182')" />

    </p>
  </body>
</html>
```

stacking.js

```
//  stacking.js
//  Illustrates dynamic stacking of images
var  top = "C172";

//  The event handler function to move the given
   element
//  to the top of the display stack
function toTop(newTop)  {


//  Set the two dom addresses, one for the old top
//  element and one for the new top element.
  domTop = document.getElementById(top).style;
  domNew = document.getElementById(newTop).style;
```

```
//  Set the zIndex properties of the two elements,
//  and reset top to the new top
  domTop.zIndex = "0";
  domNew.zIndex = "10";
  top = newTop;
}
```

Locating the Mouse Cursor

- Every event in an XHTML document creates an event object.
- A mouse-click event is an implementation of the **`MouseEvent`** interface, which defines two pairs of properties that provide coordinates for the element relative to the upper-left corner of the browser window.
  - **`ClientX`** and **`ClientY`** – coordinates within the browser window
  - **`ScreenX`** and **`ScreenY`** – coordinates within the display screen as a whole.
- The event object references in the example is created implicitly when the event occurs.

where.html

```
<!DOCTYPE html>

<!-- where.html
     Uses where.js
     Illustrates x and y coordinates of the mouse
     cursor
     -->

<html lang = "en">
  <head>
    <title> Where is the cursor? </title>
    <meta charset = "utf-8" />
    <script type = "text/javascript"
            src = "where.js">
    </script>
  </head>
  <body onclick = "findIt(event)">
    <form>
      <p>
        Within the client area: <br />
        x:
        <input type = "text" id = "xcoor1"
                             size = "4" />
        y:
        <input type = "text" id = "ycoor1"
                             size = "4" />
        <br /> <br />
        Relative to the origin of the screen
        coordinate system:
        <br />
        x:
        <input type = "text" id = "xcoor2"
                             size = "4" />
```

```
        y:
        <input type = "text" id = "ycoor2"
                             size = "4" />
      </p>
    </form>
    <p>
    </p>
      <img src = "c172.jpg" alt = "(Picture of a
  C172)" />
  </body>
</html>
```

where.js

```
//  where.js
//     Show the coordinates of the mouse cursor
//     position in an image and anywhere on the
//     screen when the mouse is clicked

//  The event handler function to get and display
//  the coordinates of the cursor, bth in an element
//  and on the screen.

function findIt(evt)  {
  document.getElementById("xcoor1").value
                                = evt.clientX;
  document.getElementById("ycoor1").value
                                = evt.clientY;
```

```
  document.getElementById("xcoor2").value
                                   = evt.screenX;
  document.getElementById("ycoor2").value
                                   = evt.screenY;
}
```

Reacting to the Mouse Click

- **`mousedown`** is the event that occurs when the left button on the mouse is pressed.
- **`mouseup`** is the event that occurs when the left button on the mouse is released.
- In the example, they are used to display and then the message, whose left and top positions are -130 and -25 to center the message around the spot where the mouse is pointing.

`anywhere.html`

```
<!DOCTYPE html>

<!-- anywhere.html
     Uses anywhere.js
     Display a message when the mouse button is
  pressed,
     no matter where it is onthe screen
     -->

<html lang = "en">
  <head>
    <title> Sense events anywhere </title>
    <meta charset = "utf-8" />
    <script type = "text/javascript" src =
  "anywhere.js">
    </script>
  </head>
```

```
   <body onmousedown = "displayIt(event)"
         onmouseup = "hideIt();">
     <p>
       <span id = "message"
             style = "color: red; visibility: hidden;
                      position: relative;
                      font-size: 1.7em;
                      font-style: italic;
                      font-weight: bold;">
         Please don't click here!
       </span>
       <br /> <br /> <br /> <br /> <br /> <br />
       <br /> <br /> <br /> <br /> <br /> <br />
       <br /> <br /> <br /> <br /> <br /> <br />
       <br /> <br /> <br /> <br /> <br /> <br />
       <br /> <br /> <br /> <br /> <br /> <br />
       <br /> <br /> <br /> <br /> <br /> <br />
       <br /> <br /> <br /> <br /> <br /> <br />
```

```
       <br /> <br /> <br /> <br /> <br /> <br />
     </p>
   </body>
 </html>
```

anywhere.js

```
//  anywhere.js
//    Display a message when the mouse button is
//    pressed, no matter where it is on the screen


//  The event handler function to display the
//  message
function displayIt(evt)  {
  var dom = document.getElementById("message");

  dom.style.left = (evt.clientX - 130) + "px";
  dom.style.top = (evt.clientY - 25) + "px";
  dom.style.visibility = "visible";
}
```

```
/***************************************************/

//  The event handler function to hide the message

function hideIt()  {
  document.getElementById
       ("message").style.visibility = "hidden";
}
```

Slow Movement of Elements

- If we want to move an item slowly across the screen, it has to be done is small increments.
- **`setTimeout`** is a method that takes two parameters: a string of JavaScript to be executed after a delay and the delay expressed in milliseconds.
  - `setTimeout("mover()", 20);`
- **`setInterval`** takes the same parameters as setTimeout (although the parameters of the JavaScript function being called can be passed as additional parameters. It performs the JavaScript code repreatedly.

`moveText.html`

```
<!DOCTYPE html>

<!-- moveText.html
     Uses moveTextfuns.js
     Illustrates a moving text element
     -->
<html lang = "en">
  <head>
    <title> Moving text </title>
    <meta charset = "utf-8" />
    <script type = "text/javascript"
            src = "moveText.js">
    </script>
  </head>
  <!-- Call the initializing function on load,
       giving the destination coordinates for the
       text to be moved
       -->

  <body onload = "initText()">
    <!-- The text to be moved, including its initial
         position
         -->
    <p>
      <span id = 'theText'
            style = "position: absolute;
            left: 100px; top: 100px;
            font: bold 1.7em 'Times Roman';
            color: blue; "> Jump in the lake!
      </span>
    </p>
  </body>
</html>
```

moveTextfuns.js

```
// This is moveTextfuns.js - used with moveText.html
   var dom, x, y, finalx = 300, finaly = 300;

//*********************************************** //

//  A function to initialize the x and y coordinates
//  of the current position of the text to be moved,
//  and then call the mover function

function initText()  {
  dom = document.getElementById('theText').style;

  /* Get the current position of the text  */
  var x = dom.left;
  var y = dom.top;
  /* Convert the string values of left and top to
     numbers by stripping off the units */
  x = x.match(/\d+/);
  y = y.match(/\d+/);

  /* Call the function that moves it */
  moveText(x, y);
}

//********************************************** //

// A function to move the text from its original
// position to (finalx, finaly)
function moveText(x, y)  {
  /* If the x coordinates are not equal, move
     x toward final x */
     if (x != finalx) x--;
     else if (x < finalx) x++;
```

```
  /* If the y coordinates are not equal, move
     x toward final x */
     if (y != finaly) y--;
     else if (y < finaly) y++;

  /* As long as the text is not at the destination,
     call the mover with the current position  */
     if ((x != finalx) || (y != finaly)) {

     /* Put the units back on the coordinates before
        assigning them to the properties to cause
        the move */
       dom.left = x + "px";
       dom.top = y + "px";
       /* Recursive call, after a 1-millimeter
          delay */
       setTimeout("moveText(" + x + ", " + y + ")",
                  1);
     }
}
```

Dragging and Dropping Elements

- Drag and drop is one of the most impressive effects of GUIs. It can be implemented using the **`mouseup`**, **`mousedown`** and **`mousemove`** events.
- The following example uses DOM 0 to call the handler for **`mousedown`**. **`grabber`**, the event handler, takes the **`Event`** object as a parameter and makes it a global variable for all the handlers to be able to use.
- It then determines the coordinates of the current position of the element to be moved and compares it to the position of the mouse cursor. The differences between these coordinates sets is used to move the element.

`dragNDrop.html`

```
<!DOCTYPE html>

<!-- dragNDrop.html
    An example to illustrate the DOM 2 Event mode.
    Allows the user to drag and drop words to
    complete a short poem
    Does not work with IE7
  -->

<html lang = "en">
  <head>
    <title> Drag and drop </title>
    <meta charset = "utf-8" />
    <script type = "text/javascript"
            src = "dragNDrop.js">
    </script>
  </head>
```

```
<body style = "font-size: 1.25em;">
  <p>
    Roses are red <br />
    Violets are blue <br />

    <span style = "position: absolute; top: 200px;
                  left: 0px;
                  background-color: lightgrey;"
                  onmousedown = "grabber(event);">
     candy </span>

    <span style = "position: absolute; top: 200px;
                  left: 75px;
                  background-color: lightgrey;"
                  onmousedown = "grabber(event);">
     cats </span>
```

```
    <span style = "position: absolute; top: 200px;
                  left: 150px;
                  background-color: lightgrey;"
                  onmousedown = "grabber(event);">
      cows </span>

    <span style = "position: absolute; top: 200px;
                  left:225px;
                  background-color: lightgrey;"
                  onmousedown = "grabber(event);">
      glue </span>

    <span style = "position: absolute; top: 200px;
                  left: 300px;
                  background-color: lightgrey;"
                  onmousedown = "grabber(event);">
      is </span>
```

```
<span style = "position: absolute; top: 200px;
              left: 375px;
              background-color: lightgrey;"
              onmousedown = "grabber(event);">
  is </span>

<span style = "position: absolute; top: 200px;
              left: 450px;
              background-color: lightgrey;"
              onmousedown = "grabber(event);">
  meow </span>

<span style = "position: absolute; top: 250px;
              left: 0px;
              background-color: lightgrey;"
              onmousedown = "grabber(event);">
  mine </span>
```

```
<span style = "position: absolute; top: 250px;
              left: 75px;
              background-color: lightgrey;"
              onmousedown = "grabber(event);">
  moo </span>

<span style = "position: absolute; top: 250px;
              left: 150px;
              background-color: lightgrey;"
              onmousedown = "grabber(event);">
  new </span>

<span style = "position: absolute; top: 250px;
              left: 225px;
              background-color: lightgrey;"
              onmousedown = "grabber(event);">
  old </span>
```

```
<span style = "position: absolute; top: 250px;
              left: 300px;
              background-color: lightgrey;"
              onmousedown = "grabber(event);">
  say </span>

<span style = "position: absolute; top: 250px;
              left: 375px;
              background-color: lightgrey;"
              onmousedown = "grabber(event);">
  say </span>

<span style = "position: absolute; top: 250px;
              left: 450px;
              background-color: lightgrey;"
              onmousedown = "grabber(event);">
  so </span>
```

```
<span style = "position: absolute; top: 300px;
              left: 0px;
              background-color: lightgrey;"
              onmousedown = "grabber(event);">
  sticky</span>

<span style = "position: absolute; top: 300px;
              left: 75px;
              background-color: lightgrey;"
              onmousedown = "grabber(event);">
  sweet </span>

<span style = "position: absolute; top: 300px;
              left: 150px;
              background-color: lightgrey;"
              onmousedown = "grabber(event);">
  syrup </span>
```

```
        <span style = "position: absolute; top: 300px;
                      left: 225px;
                      background-color: lightgrey;"
                      onmousedown = "grabber(event);">
          too </span>

        <span style = "position: absolute; top: 300px;
                      left: 300px;
                      background-color: lightgrey;"
                      onmousedown = "grabber(event);">
          yours </span>

      </p>
    </body>
  </html>
```

dragNDrop.js

```
//  dragNDrop.js
//  An example to illustrate the DOM 2 Event mode
//  Allows the user the drag and drop words to
  complete
//  a short poem
//  Does not work with IE7

//  Define variables for the values computed by
//  the grabber vent handler but needed by mover
//  event handler

    var diffX, diffY, theElement;

// ***********************************************
// The event handler function for grabbing the word
function grabber(event)  {

  //  Set the global variable for the element to be
  //  moved
  theElement = event.currentTarget;

  //  Determine the position of the word to be
  //  grabbed, first removing the units from left
  //  and top
  var posX = parseInt(theElement.style.left);
  var posY = parseInt(theElement.style.top);

  //  Compute the difference between where it is and
  //  where the mouse click occurred
  diffX = event.clientX - posX;
  diffY = event.clientY - posY;
```

```
  //  Now register the event handlers for moving and
  //  dropping the word
  document.addEventListener("mousemove",
                                   mover, true);
  document.addEventListener("mouseup",
                                   dropper, true);

  //  Stop propagation of the event and stop any
  default
  //  browser action
  event.stopPropagation();
  event.preventDefault();
}

//*************************************************
```

```
//  The event handler function for moving the word
function mover(event)  {

  //  Compute the new position, add the units and
  //  move the word
  theElement.style.left
            = (event.clientX - diffX) + "px";
  theElement.style.top
            = (event.clientY - diffY) + "px";

  //  Prevent propagation of the event
  event.stopPropagation();
}


//**************************************************
```

```
// The event handler function for dropping the word
function dropper(event)  {
  //  Unregister the event handlers for mouseup and
  //  mousemove
  document.removeEventListener("mouseup", dropper,
   true);
  document.removeEventListener("mousemove", mover,
   true);

  // Prevent propagation of the event
  event.stopPropagation();
}
```