Dynamic Documents

with JavaScript

Dynamic XHTML documents

• Πρόκειται για αρχεία XHTML τα οποία μπορούν να αλλάζουν κατά τη διάρκεια της παρουσίασής τους στο παράθυρο ενός πληκτρολογίου.
• Ο ποιό κοινός τρόπος υλοποίησης δυναμικών εγγράφων είναι με χρήση JavaScript για επεξεργασία αντικειμένων του DOM του εμφανιζόμενου εγγράφου.
• Αλλαγές σε ένα έγγραφο μπορούν να συμβούν:
  – Ως αποτέλεσμα αλληλεπιδράσεων του χρήστη με τον πληκτρολόγιο
  – Σε τακτά χρονικά διαστήματα
  – Όταν προκύπτουν κάποια συμβάντα στον πληκτρολόγιο
• Def: A dynamic HTML document is one whose tag attributes, tag contents, or element style properties can be changed after the document has been and is still being displayed by a browser

Positioning Elements

• Στις αρχικές εκδόσεις της HTML και των πληκτρολόγων, η στοιχειοθεσία των στοιχείων HTML στην οθόνη των πληκτρολόγων ήταν απλή - ο προγραμματιστής είχε μικρή επιρροή στην παρουσίαση/τοποθέτηση των HTML στοιχείων στην οθόνη:
  – Πίνακες, Πλαίσια (frames)
• Cascading Style Sheets-Positioning (CSS-P):
  – Ανακοινώθηκαν από το W3C το 1997
  – Πλήρης υποστήριξη από IE8, FX3
Absolute positioning

- The **absolute** value is specified for position when the element is to be placed at a specific place in the document display without regard to the positions of other elements.
  - One use is to superimpose special text over a paragraph of ordinary text to create an effect similar to a watermark on paper
  - If an element is **nested** inside another element and is absolutely positioned, the top and left properties are relative to the enclosing element

```
<p style = "position: absolute; left: 50px; top: 100px;">
```

Apple is the common name for any tree of the genus Malus, of the family Rosaceae. Apple trees grow in any of the temperate zones of the world. Some apple blossoms are fragrant; some apple blossoms emit a non-fragrant, somewhat musty and disagreeable odour. The apple tree is deciduous, losing its leaves in autumn. The wood of the apple tree is fine-grained and hard. It is, therefore, good for furniture and cabinetwork. Apple trees have been from the earliest centuries. They are propagated by grafting because they do not reproduce themselves.
Relative positioning

- If no top and left properties are specified, the element is placed exactly where it would have been placed if no position property were given
  - But it can be moved later
- If top and left properties are given, they are offsets from where it would have placed without the position property being specified

Static positioning

- The default value for the position property is static.
- A statically positioned element is placed in the document as if it had the position value of relative, but no top or left were given.
- The difference is that a statically positioned element cannot have its top or left properties initially set or changed later.
  - Therefore, a statically placed element cannot be displaced from its normal position and cannot be moved from that position later.
6.3 Moving Elements

- If `position` is set to either `absolute` or `relative`, the element can be moved after it is displayed.
- Just change the `top` and `left` property values with a script.

→ SHOW mover.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";
}
6.4 Element Visibility

- The visibility property of an element controls whether it is displayed

- The values are visible and hidden

```javascript
if (dom.visibility == "visible")
    dom.visibility = "hidden";
else
    dom.visibility = "visible";
```

→ SHOW showHide.html & showHide.js
6.5 Changing Colors and Fonts

- Background color is controlled by the `backgroundColor` property
- Foreground color is controlled by the `color` property

Background color:

```html
<input type = "text" size = "10"
name = "background"
onchange = "setColor('background',
this.value)">
```

- The actual parameter `this.value` works because at the time of the call, `this` is a reference to the text box (the element in which the call is made)
- So, `this.value` is the name of the new color

→ SHOW dynColors.html & dynColors.js
6.5 Dynamic Colors and Fonts (continued)

- Changing fonts
  - We can change the font properties of any element that contains text by using the `mouseover` and `mouseout` events to trigger a script that makes the changes
  ```javascript
  onmouseover = "this.style.color = 'blue';
  this.style.font = 'italic 16pt Times';"
  onmouseout = "this.style.color = 'black';
  this.style.font = 'normal 16pt Times';"
  ```

- JavaScript property names:
  - For CSS attributes w/o hyphens – same
  - For CSS attributes w/ hyphens – delete hyphen and capitalize the next letter – `font-size` -> `fontSize`

SHOW `dynFont`
6.6 Dynamic Content

- The content of an HTML element is addressed with the `value` property of its associated JavaScript object
- Example: a help box for a form

SHOW `dynValue.html` & `dynValue.js`

---

```html
<!doctype html>
<html>
  
  <head>
    
    <title>DynValue</title>
    <script type="text/javascript" src="dynValue.js"></script>
  
  
  </head>

  <body>
    
    Customer information
    <form action="">
      
      <label>
        Name: 
        <input type="text" onmouseover="messages(0)"
               onmouseout="messages(4)"
               style="position: absolute; left: 45px; top: 0px"> 
      </label>
      
      Email:  
      <input type="text" onmouseover="messages(1)"
             onmouseout="messages(4)"
             style="position: absolute; left: 45px; top: 0px"> 
      
      <textarea id="adviceBox" rows="3" cols="50"
                style="position: absolute; left: 45px; top: 0px"> 
      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 />
      
      <input type="submit" value="Submit" />
      
      <input type="reset" value="Reset" />
      
    </form>

  </body>

</html>
```

---

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

var helpers =eturad {"Your name must be in the form: \n
first name, middle initial, last name",
"Your email address must have the form: \nuserDomain",
"Your user ID must have at least six characters",
"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];
}
```
6.7 Stacking Elements

- The **z-index** attribute determines which element is in front and which are covered by the front element.

- The JavaScript property associated with the **z-index** attribute is `zIndex`.

- **z-index** can be changed dynamically (by changing `zIndex`).

- An image element can have an `onclick` attribute, so images can be clicked to trigger event handlers.

- Anchors can also trigger event handlers when they are clicked.

- The `href` attribute can be set to call a JavaScript function by assigning it the call, with `'JAVASCRIPT'` attached to the call code:

  ```html
  <a href = "JAVASCRIPT:fun()">
  </a>
  ```

6.7. Stacking Elements (continued)

- To change stacking order, the handler function must change the `zIndex` property value of the element.

- A call to the function from an element sets the `zIndex` value of the new top element to 10 and the `zIndex` value of the old top element to 0.

- It also sets the current top to the new top.

→ SHOW `stacking.html` & `stacking.js`

```javascript
// stacking.js
var top = "C172";

function toTop(newTop) {
  domTop = document.getElementById(top).style;
  domNew = document.getElementById(newTop).style;

  domTop.zIndex = "0";
  domNew.zIndex = "10";
  top = newTop;
}
```
6.8 Locating the Mouse Cursor

- Every event that occurs while an XHTML document is being displayed creates an event object. This object includes some information about the event.
- A mouse click event defines two pairs of properties that provide geometric coordinates of the position of the element in the display that created the event:
  - The coordinates of the element that caused an event are available in the `clientX` and `clientY` properties of the `event` object
    - These are relative to upper left corner of the browser display window
  - `screenX` and `screenY` are relative to the upper left corner of the whole client screen
- If we want to locate the mouse cursor when the mouse button is clicked, we can use the `click` event
6.9 Reacting to a Mouse Click

- A mouse click can be used to trigger an action, no matter where the mouse cursor is in the display

- Use event handlers for `onmousedown` and `onmouseup` that change the visibility attribute of the message

→ SHOW `anywhere.html` & `anywhere.js`
6.10 Slow Movement of Elements

- To animate an element, it must be moved by small amounts, many times, in rapid succession
- JavaScript has two ways to do this, but we cover just one: setTimeout("fun()"), n
- Example: move a text element from its initial position (100, 100) to a new position (300, 300)
  - Use the onload attribute of the body element to initialize the position of the element
  - Use a move function to change the top and left attributes by one pixel in the direction of the destination
  - A problem: coordinate properties are stored as strings, which include the units ("150px")
  - Another problem: We need to use some HTML special characters ("<" and "-->"

1. XML parsers may remove all comments
2. Put the script in a CDATA section
3. Put JavaScript in separate file

- These are problems of validation only
6.11 Dragging and Dropping an Element

- We can use mousedown,mousemove, and mouseup events to grab, drag, and drop
- We know how to move an element - just change its left and top properties
- Example: magnetic poetry
- The DOM 2 event model is required (the Event object and its property, currentTarget)
- We use both DOM 0 and DOM 2 models (DOM 0 to call the mousedown handler, grabber)

```javascript
function moveText(x, y) {
    /* If the x coordinates are not equal, move x toward finalx */
    if (x != finalx) {
        if (x > finalx) x--;
        else if (x < finalx) x++;
    }
    /* If the y coordinates are not equal, move y toward finaly */
    if (y != finaly) {
        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-millisecond delay */
        setTimeout('moveText("' + x + "," + y + ")', 1);
    }
}
```

SHOW dragNDrop.html & dragNDrop.js