

Dynamic Documents with JavaScript

Objectives

- Develop dynamic HTML documents
- Using CSS-P for controlling element position
- Dynamically changing element visibility, color, fonts, or content
- Using the zIndex to control element stacking
- Handling mouse events
- Implementing drag and drop functionality

Introduction

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

For example, a dynamic document may change the color, font, or position of an HTML element after a mouse click.

Controlling Position

Positioning Elements

Changing the position of an HTML element was made easier by positioning extensions to CSS (CSS-P)

The extension was released by W3C in 1997 and is now supported by all modern browsers

CSS-P allowed the position of any element to be specified by the three style properties: **position**, **left**, and **top**

The positioning property decides how the elements will be laid out. The three possible values of position are **absolute**, **relative**, and **static**

The left and top properties control the placement position depending on the position property.

Absolute Positioning

Absolute positioning specifies placement of elements relative to the enclosing element

The ***enclosing element*** is the element in which the given one is nested. This could be a <div>, a <p>, or some other tag, or could even be the whole document.

To state the position of the element, set the left and top properties to needed pixel values, e.g.:

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

Relative Positioning

Relative positioning places HTML elements relative to other elements

If `top` and `left` properties are given, they are offsets from where the element would have placed without the `position` property being specified

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!**

Static Positioning

Static positioning is the default value of position if `position` is not specified

A statically placed element **cannot be dynamically moved** from its position

So to move elements, set the `position` to either `absolute` or `relative`. The element can be moved after it is displayed by changing the `top` and `left` property values using JavaScript

Slow Movement of Elements

We can **animate an element** by changing its left and top properties by small amounts, many times, in rapid succession.

JavaScript has two ways to do this, but we cover just one: using the `setTimeout` method

The **`setTimeout`** method sets a timer that triggers a call to the specified function after a specified number of milliseconds has passed, e.g.:

```
setTimeout ("makeChange () ", 5) ;
```

calls the function `makeChange ()` after 5 ms.

Slow Movement of Elements (cont.)

We can hence use `setTimeout` to call a function which will make tiny changes to position

Example

To 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 and call a *move function*
- Use a *move function* to change the `top` and `left` attributes by one pixel in the direction of the destination
- Before ending the move function, call `setTimeout` to repeat the *move function* again in 1 ms

One problem: coordinate properties are stored as strings, which include the units ("150px")

Examples

absPos.html [\[link\]](#)

absPos2.html [\[link\]](#)

relPos.html [\[link\]](#)

mover.html [\[link\]](#)

moveText.html [\[link\]](#)

Start Session 19

Controlling Visibility, Colors, Fonts, and Content

Element Visibility

Another important aspect we can control within a script is **element visibility**

This can be done by setting the **visibility** property. Its two values are **visible** and **hidden**.

For example, we can have code that toggles the visibility after some event occurs:

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

Changing Colors

We also change the element's background or foreground color, e.g.:

```
document.body.style.backgroundColor =  
    newColor;
```

Note that in JavaScript, foreground color property is just **color**, but background color is **backgroundColor**, not `background-color` as it is in CSS.

In general, **JavaScript property names** follow these rules:

- For CSS attributes w/o hyphens – same name
- For CSS attributes w/hyphens – **delete hyphen and capitalize the next letter** – `font-size` -> `fontSize`

Changing Fonts

Here is another dynamic page example:

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, e.g.:

```
onmouseover = "this.style.color = 'blue';  
               this.style.font = 'italic 16pt Times';"
```

```
onmouseout = "this.style.color = 'black';  
              this.style.font = 'normal 16pt Times';"
```


Dynamic Content

The **content** of an HTML element is addressed with the **value** property of its associated JavaScript object

This is useful for getting the text data from a textbox

Example: a help box for a form

Examples

showHide.html [\[link\]](#)

dynColors.html [\[link\]](#)

dynFont.html [\[link\]](#)

dynValue.html [\[link\]](#)

Element Stacking

Stacking Elements

The CSS **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`, which can be changed dynamically

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

Higher `zIndex` values will make the elements go on top

Examples

stacking.html [\[link\]](#)

Mouse Cursor and Events

Locating the Mouse Cursor

If we want to locate the mouse cursor when the mouse button is clicked, we can use the `click` event

The coordinates of the element that caused an event (e.g. mouse click) are available in the **`clientX`** and **`clientY`** properties of the `event` object

- These are relative to upper left corner of the browser display window

There exist also **`screenX`** and **`screenY`** properties, which are relative to the upper left corner of the whole client screen

Note that a mouse click can be used to trigger an action, no matter where the mouse cursor is in the display

Examples

where.html [\[link\]](#)

anywhere.html [\[link\]](#)

Implementing Drag and Drop

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

So to implement drag and drop, we can use the DOM 2 event model to **add an event handler** for `mousemove` whenever a `mousedown` event occurs and **remove it** whenever a `mouseup` event occurs afterwards

We will know which element to move by accessing the `Event` object and its property, `currentTarget`

Drag and Drop

So here is the process:

- When the `mousedown` event occurs, get a reference of the element to be moved when the mouse button is pressed down (using `currentTarget`) and register events for `mousemove` and `mouseup`
- When the `mousemove` event occurs, move the element by changing its `top` and `left` properties of the element as the mouse cursor is moved
 - Compute the distance of each move as the difference between the current position (the `left` and `top` values) and the mouse click position (`clientX` and `clientY`)
- When the `mouseup` event occurs, drop the element by removing the event handler for `mousemove` and `mouseup`

Examples

dragNDrop.html [\[link\]](#)

Summary

- 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
- CSS-P allows the position of any element to be specified by the three style properties: position, left, and top
- The style property of an HTML element can be changed within JavaScript to dynamically control color, fonts, and other aspects of the element's presentation
- The zIndex is a property that controls the order in which elements are drawn to the screen
- Mouse coordinates can be accessed through the mouse event object using either the clientX and clientY, or screenX and screenY properties
- Drag and drop functionality can be implemented by dynamically adding and removing event handlers using the DOM 2 event model