Web Technology & Programming

DHTML

LECTURER: KRUNAL PATEL
ICCT ENGG. COLLEGE
NEW VALLABH VIDYANAGAR

Introduction

- Everyone is a Web Designer.
  - Learning DHTML, CSS and JavaScript is your next step into the web design world.
- "***DHTML***" is a term used by some vendors to describe the combination of ***HTML***, ***style sheets*** and ***scripts*** that allows documents to be animated.
- Web pages can be made to respond to user's actions.
- Problem: How to achieve "Dynamic"?

Introduction

- As discussed earlier ***DHTML*** is the combination of 3 browser technologies to render a dynamic browser experience
  - HTML for content
  - Cascading Style Sheets for general presentation
  - Javascript to enable dynamic presentation
- In reality it is tapping into the richness of the browser's Document Object Model (DOM) using HTML, CSS and Javascript that is the key to a highly dynamic browser experience

Introduction

- With Dynamic HTML the emphasis is on making the browser provide a richer viewing experience through effects like
  - Animation
  - Filters
- This is done by applying properties and methods of the Browser's Document Object Model (DOM)
- This is usually done through Javascript, but it can also be done by any scripting language supported by the browser, such as VB-Script

DHTML advantages

- Supported by most browsers
- Small file sizes (faster than Flash)
- No plug-ins required
- Easy to learn (learn HTML, JavaScript)
- Faster web experience (change the page content without load new pages)
- Browser and OS incompatibilities
  - The implementation of CSS, DOM varies from browser to browser

DHTML Disadvantages

- Time-consuming, cross-platform de-bugging necessary
- won't run on older browsers

The Role of each component in DHTML

- ***With CSS***, we can change the style of any HTML elements.
- ***With DOM,*** we can have a map on every elements in the HTML page.
- ***With JavaScript,*** we can access and have operations on the elements in the DOM tree.
- ***With event handler,*** we can execute the predefined scripts at any time.

The Role of each component in DHTML

With HTML

```
<html>
<body>

<h1 id="header">Example</h1>

<script type="text/javascript">
document.getElementById("header").innerHTML="First DHTML
   example";
</script>

</body>
</html>
```

The Role of each component in DHTML

With CSS & JavaScript

```
<html>
<body>

<h1 onclick="this.style.color='red'">Click</h1>

</body>
</html>
```

The Role of each component in DHTML

With Event

```
<html>
<head>
        <script type="text/javascript">
                function change(id)
                {
                        id.innerHTML="Text changed";
                }
        </script>
  </head>
  <body>
        <h1 onclick="change(this);">Click on this text</h1>
  </body>
</html>
```

Animation

- With JavaScript, it is possible to execute some code after a specified time-interval. This is called ***Timing Events***.
- It's very easy to time events in JavaScript.
- The two key methods that are used :
  - **setTimeout()** - executes a code some time in the future
  - **clearTimeout()** - cancel the setTimeout()

Animation

The setTimeout() Method

Syntax

var t=setTimeout("*javascript statement*",*milliseconds*);

- The setTimeout() method returns a value. In the syntax defined above, the value is stored in a variable called t. If you want to cancel the setTimeout() function, you can refer to it using the variable name.
- The first parameter of setTimeout() can be a string of executable code, or a call to a function.
- The second parameter indicates how many milliseconds from now you want to execute the first parameter.
- **Note:** There are 1000 milliseconds in one second.

Events

- An E***vent*** is defined as "***something that takes place***" and that is exactly what it means in web programming as well.
- An event handler is JavaScript code that is designed to run each time a particular event occurs.
- Syntax of handling the events
  - <Tag Attributes **event="handler"**>

Categories of Events

- Events fall into four major categories:
  - User interface events
  - Mouse events
  - Key events
  - HTML events
- **User interface** events happen as controls or other objects on a web page gain and lose focus. These events are often caused by other user actions such as a tab key press. They can also happen programmatically as well.
- **Mouse events** occur when the user moves the mouse or presses one of the mouse buttons. These events allow a web page to respond to mouse movements by.
- **Key events** occur when the user presses and/or releases one of the keyboard keys. Only certain HTML elements can capture keyboard events.
- Finally, there are several events specific to certain **HTML** elements. They often relate to the browser window itself or to form controls and other objects embedded in a web page.

Handle User Interface Events

- User interface events deal exclusively with the transfer of focus from one object inside the web page to another. There are three user interface events defined in most web browsers.

Event Name	Event Handler Name
focus	onfocus
blur	onblur
activate	onactivate

Example

```
<script type="text/javascript" language="javascript">
    function upd(instr)
    {
            document.forms[0].statusbox.value += instr + "; ";
    }
</script>

<form action="#">
    <input type="text" name="box1“ onblur="upd('blur box1')"><br>
    <input type="text" name="box2“ onfocus="upd('focus box2')“
    onactivate="upd('activate box2')"><br><br>
            Event firing order:
    <input type="text" name="statusbox" size="40">
</form>
```

Handle Mouse Events

- JavaScript programs have the ability to track mouse movement and button clicks as they relate to the web page and controls inside

Event Name	Event Handler Name
mousedown	onmousedown
mouseup	onmouseup
mouseover	onmouseover
mousemove	onmousemove
mouseout	onmouseout
click	onclick
dblclick	ondblclick

Example

```
<script type="text/javascript" language="javascript">
function changeimage(num) {
var img = document.getElementById("SampleImage");
if (num == 1) {
img.src = “img.gif";
} else {
img.src = “flower.gif";
}
}
</script>
<img src=“img.gif"
id="SampleImage"
alt="Sample image"
onmouseover="changeimage(1)"
onmouseout="changeimage(2)" >
```

Example

```
<script type="text/javascript" language="javascript">
function upd(instr) {
document.forms[0].statusbox.value += instr + "; ";
}
</script>
<form action="#">
<input type="text" name="box1"
onkeypress="upd('keypress')"
onkeydown="upd('keydown')"
onkeyup="upd('keyup')"><br><br>
Event firing order:
<input type="text" name="statusbox" size="40">
</form>
```

Handle HTML Events

- HTML events means any events that do not belong in the user interface, mouse, or key event categories.
- Some HTML events are triggered directly by a user action, while others are fired only as an indirect result of a user action.

List of HTML Events

Event Name	Event Handler Name	Fires When (Event)
load	onload	Browser finishes loading document
unload	onunload	Browser about to unload document
submit	onsubmit	Form about to be submitted
reset	onreset	Form about to be reset
select	onselect	Text box contents selected
change	onchange	Form control contents changed
abort	onabort	User aborts download of image
error	onerror	Error occurs on object loading
resize	onresize	Size of object about to change
scroll	onscroll	User uses the scroll bar

Event handler	Applies to:
onAbort	Image
onBlur	Button, Checkbox, FileUpload, Layer, Password, Radio, Reset, Select, Submit, Text, TextArea, Window
onChange	FileUpload, Select, Text, TextArea
onClick	Button, Document, Checkbox, Link, Radio, Reset, Submit
onDblClick	Document, Link
onDragDrop	Window
onError	Image, Window
onFocus	Button, Checkbox, FileUpload, Layer, Password, Radio, Reset, Select, Submit, Text, TextArea, Window
onKeyDown	Document, Image, Link, TextArea
onKeyPress	Document, Image, Link, TextArea
onKeyUp	Document, Image, Link, TextArea
onLoad	Image, Window
onMouseDown	Button, Document, Link
onMouseOut	Image, Link
onMouseOver	Image, Link
onMouseUp	Button, Document, Link
onMove	Window
onReset	Form
onResize	Window
onSelect	Text, Textarea
onSubmit	Form
onUnload	Window