*JavaScript and AJAX*
2nd Edition

# Tutorial 4

Working with Objects and Styles

## Objectives

- Learn about objects and the document object model
- Reference documents objects by ID, name, and tag name
- Work with object collections
- Work with object properties and CSS styles
- Study the syntax of object methods
- Apply an event handler to an object

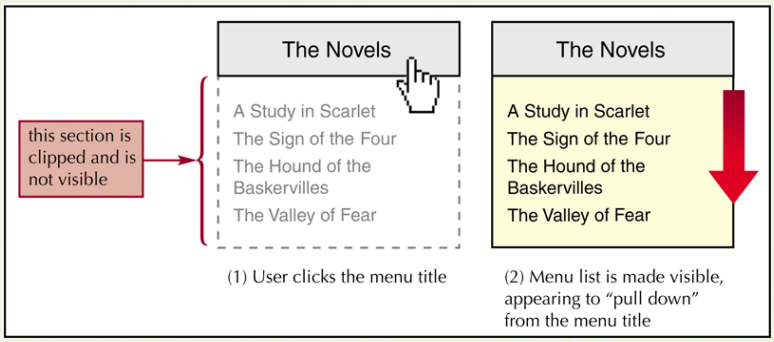
# Objectives (cont'd)

- Work with mouse events
- Employ object detection to avoid programming errors
- Create an animation with timed functions
- Explore how to create sliding and tabbed menus
- Retrieve values from a style sheet
- Create custom objects



# Introducing Pull-Down Menus

- In a **pull-down menu**, a menu title is always visible to the user, identifying the entries in the menu



# Introducing Objects, Properties, Methods, and Events

- JavaScript
  - **Object-based** language
    - Based on manipulating objects through use of **properties, methods,** and **events**
  - Supports three types of objects:
    - **Built-in objects**
    - **Document objects**
    - **Customized objects**



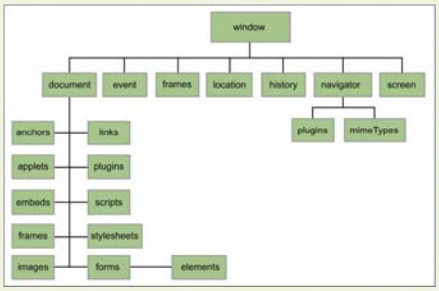
# Exploring the Document Object Model

- **Document object model (DOM)**
  - Organized structure of objects
  - Goal is to make every object related to the document or Web browser accessible to a scripting language

| DOM | Description |
|---|---|
| DOM Level 0 (Basic Model) | The basic DOM that supported few page and browser objects and allowed dynamic content only for form elements |
| DOM Level 0 + Images | The basic DOM with added support for image rollovers |
| Netscape 4 (layers) | The basic DOM with support for the Netscape 4 layer element and the ability to capture events within the browser |
| Internet Explorer 4 | An expanded DOM allowing dynamic content for most page elements |
| Internet Explorer 5 | The IE 4 DOM with additional refinements and enhancements |
| W3C DOM Level 1 | The first DOM specification by the W3C, which supported all page and browser elements and handled all events occurring within the browser |
| W3C DOM Level 2 | The second DOM specification, allowing for the capture of events, manipulation of CSS styles, working with element text, and document subsets |
| W3C DOM Level 3 | The third DOM specification, providing a framework for working with document loading and saving, as well as working with DTDs and document validation |

# Exploring the Document Object Model

- Each document object model organizes objects into a hierarchy known as a **document tree**
- Tree structure becomes more elaborate as DOMs encompass more objects






# Referencing Objects

- **Object name** identifies each object
- Use **dot syntax** (i.e., separate each level using a dot) to indicate location of an object within the hierarchy

| Object Name | Description |
|---|---|
| window | The browser window |
| document | The Web document displayed in the window |
| document.body | The body of the Web document displayed in the browser window |
| event | Events or actions occurring within the browser window |
| history | The list of previously visited Web sites within the browser window |
| location | The URL of the document currently displayed in the browser window |
| navigator | The browser itself |
| screen | The screen displaying the document |

# Referencing Objects

- **Object collections**
  - Are arrays of more than one of the same type of object
  - Support the length property

| Object Collection | Description |
|---|---|
| document.anchors | All anchors |
| document.applets | All applets |
| document.embeds | All embed elements |
| document.forms | All Web forms |
| document.*form*.elements | All elements within a specific *form* |
| document.images | All inline images |
| document.links | All links |
| document.plugins | All plug-ins in the document |
| document.styleSheets | All style sheet elements |
| navigator.plugins | All plug-ins supported by the browser |
| navigator.mimeTypes | All mime-types supported by the browser |
| window.frames | All frames within the browser window |



# Referencing Objects

| ***To reference…*** | ***Use…*** |
|---|---|
| An object as part of the collection in a document | `collection[idref]` or `collection.idref` |
| A document object based on its ID | `document.getElementById(id)` |
| An array of elements based on the tag name | `object.getElementsByTagName(tag)` |
| An array of elements based on the value of the name attribute | `document.getElementsByName(name)` |

# Working with Object Properties

- Most objects are associated with one or more object properties

```
object.property
```

- To set the value of an object property:

```
object.property = expression
```

- To apply a CSS style to a document object:

```
object.style.attribute = expression
```



# Working with Object Properties

- Conventions for object properties that mirror HTML attributes:
  - Begin with lowercase
  - Use camel case for multiple words
  - Are prefaced with text string html if a reserved JavaScript name/keyword

# Exploring Object Methods

- To apply a method to an object:
  *object.method(parameters)*

| Expression | Action |
|---|---|
| location.reload() | Reload the current page in the browser |
| document.forms[0].reset() | Reset the first form in the Web page |
| document.forms[0].submit() | Submit the first form in the Web page |
| document.write("Sherlock Holmes Novels") | Write "Sherlock Holmes Novels" to the Web page |
| history.back() | Go back to the previous page in the browser's history list |
| thisDay.getFullYear() | Return the four-digit year value from the thisDay date object |
| Math.rand() | Return a random value using the Math object |
| navigator.javaEnabled() | Return a Boolean value indicating whether Java is enabled in the browser |
| window.close() | Close the browser window |
| window.print() | Print the contents of the browser window |
| window.scroll(*x*, *y*) | Scroll the browser window to the (*x*, *y*) coordinate |



# Working with Event Handlers

- All objects can be affected by events initiated by the user or browser

```
window.onload = init;

function init() {
   var menus = new Array();
   var allElems = document.getElementsByTagName("*");

   for (var i = 0; i < allElems.length; i++) {
      if (allElems[i].className == "menu") menus.push(allElems[i]);
   }

}
```

runs the init() function after the page is loaded into the browser window

# Treating an Event Handler as an Object Property

- Advantages:
  - Provides greater flexibility in designing scripts
  - Removes scripting from the HTML code, placing it within the external script file
- Disadvantages:
  - Difficulty passing parameter values to the function assigned to the event
  - Can assign only one function at a time to an object and event



# Programming a Pull-Down Menu

- ChangeMenu function changes the pull-down menu from one menu list to another

```
function init() {
   var menus = new Array();
   var allElems = document.getElementsByTagName("*");

   for (var i = 0; i < allElems.length; i++) {
      if (allElems[i].className == "menu") menus.push(allElems[i]);
   }

   for (var i = 0; i < menus.length; i++) {
      menus[i].onclick = changeMenu;
   }
}

function changeMenu() {
   // this function changes the pull-down menu displayed in the document
}
```

loops through the menus array, adding an onclick event handler to each menu title

initial code for the changeMenu() function

# Programming a Pull-Down Menu

- ActiveMenu variable contains object reference of the current pull-down menu

```
var activeMenu = null;

function init() {
   var menus = new Array();
   var allElems = document.getElementsByTagName("*");

   for (var i = 0; i < allElems.length; i++) {
      if (allElems[i].className == "menu") menus.push(allElems[i]);
   }

   for (var i = 0; i < menus.length; i++) {
      menus[i].onclick = changeMenu;
   }

}
```

the activeMenu variable stores the object reference of the pull-down menu



# Programming a Pull-Down Menu

- The **this** keyword references the currently active object in the Web browser

```
function changeMenu() {
   // this function changes the pull-down menu displayed in the document

   closeOldMenu();

   menuID = this.id + "List";
   activeMenu = document.getElementById(menuID);
   activeMenu.style.display = "block";
}
```

`this` keyword references the object that called the changeMenu() function

# Adding Handlers for Mouse Events

- Enables the displayed pull-down menu to change as users move the pointer from one menu title to another

```
function init() {
   var menus = new Array();
   var allElems = document.getElementsByTagName("*");

   for (var i = 0; i < allElems.length; i++) {
      if (allElems[i].className == "menu") menus.push(allElems[i]);
   }

   for (var i = 0; i < menus.length; i++) {
      menus[i].onclick = changeMenu;
      menus[i].onmouseover = moveMenu;
   }

   document.getElementById("logo").onclick = closeOldMenu;
   document.getElementById("linkList").onclick = closeOldMenu;
   document.getElementById("main").onclick = closeOldMenu;

}

function moveMenu() {
   // this function moves the pull-down menu from one title to another

}
```

runs the moveMenu() when the pointer initially moves over a menu title



# Animating a Pull-Down Menu

- Create the illusion of a menu being pulled down using the setInterval() method and the CSS clip style

# Animating a Pull-Down Menu

- Adding the rollDown() function

```
function closeOldMenu() {
   if (activeMenu) {
      activeMenu.style.display = "none";
      activeMenu = null;
   }
}

function rollDown() {
   clipHgt = clipHgt + 10;
   if (clipHgt < 400) {
      activeMenu.style.clip = "rect(0px, 150px," + clipHgt + "px, 0px)";
   } else {
      clearInterval(timeID);
      clipHgt = 0;
   }
}
```

increases the clipping height by 10 pixels

if the height is less than 400 pixels, clips the menu

otherwise stops running the rollDown() function and resets the clipHgt value to 0



# Animating a Pull-Down Menu

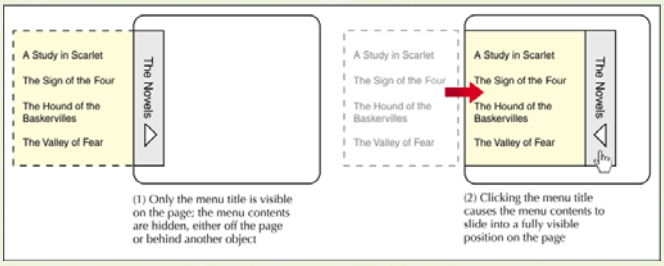
- Calling the rollDown() function

```
function moveMenu() {
   // this function moves the pull-down menu from one title to another

   if (activeMenu) {
      closeOldMenu();

      menuID = this.id + "List";
      activeMenu = document.getElementById(menuID);
      activeMenu.style.clip = "rect(0px, 150px, 0px, 0px)";
      activeMenu.style.display = "block";
      timeID = setInterval("rollDown()", 1);
   }
}

function changeMenu() {
   // this function changes the pull-down menu displayed in the document

   closeOldMenu();

   menuID = this.id + "List";
   activeMenu = document.getElementById(menuID);
   activeMenu.style.clip = "rect(0px, 150px, 0px, 0px)";
   activeMenu.style.display = "block";
   timeID = setInterval("rollDown()", 1);
}
```

runs the rollDown() function once every millisecond until it is cleared

# Creating Other Types of Menus

- **Pop-up menu:** User clicks an object on the page and the menu appears, sometimes elsewhere on the page
- **Sliding menu:** A menu is partially hidden either off the Web page or behind another object






# Creating Other Types of Menus

- **Tabbed menu:** Several menus are stacked on the page with one part of each menu visible to the user