Modern Programming Languages-JavaScript
Lecture 39

Objects

- **An object is a collection of data types as well as functions in one package**
- **The various data types called properties and functions called methods are accessed using the dot notation**

- **objectname.propertyname**

Objects

- **There are many types of objects in JavaScript**

  - **Built-in objects (primarily type related)**
  - **Browser objects (navigator, window etc.)**
  - **Document objects (forms, images etc.)**
  - **User defined objects**

Two Object Models

- **In JavaScript, two primary object models are employed**
- **Browser Object Model (BOM)**

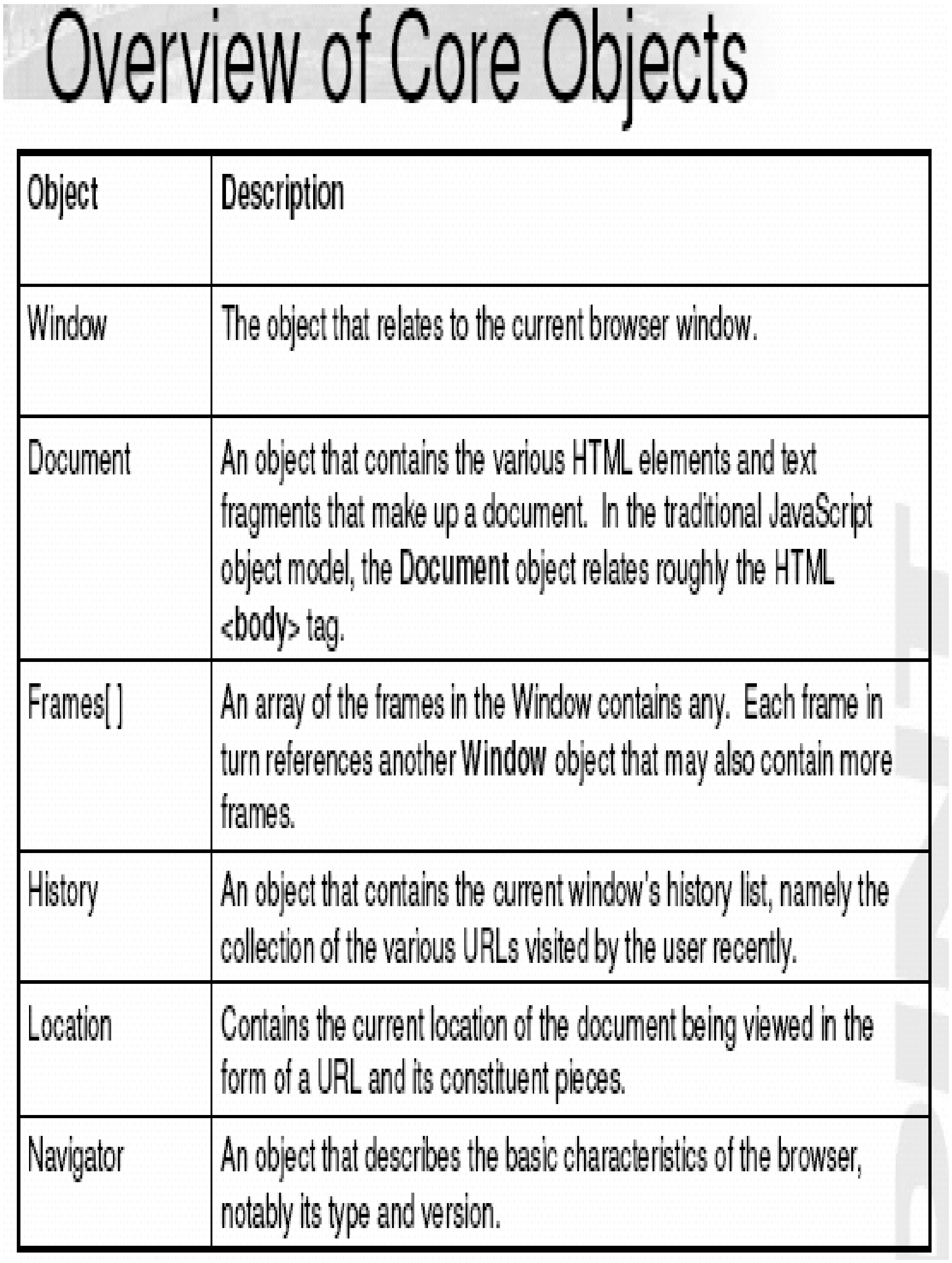
  - **The BOM provides access to the various characteristics of a browser such as the browser window itself, the screen characteristics, the browser history and so on.**
- **Document Object Model (DOM)**
  - **The DOM on the other hand provides access to the contents of the browser window, namely the documents including the various HTML elements ranging from anchors to images as well as any text that may be enclosed by such element.**

The Big Picture

- Looking at the "big picture" of all various aspects of JavaScript including its object models. We see four primary pieces:
  1. The core JavaScript language (e.g. data types, operators, statements, etc.)
  2. The core objects primarily related to data types (e.g. Date, String, Math, etc.)
  3. The browser objects (e.g. Window, Navigator, Location, etc.)
  4. The document objects (e.g. Document, Form, Image, etc.)

Four Models

- By studying the history of JavaScript we can bring some order to the chaos of competing object models. There have been four distinct object models used in JavaScript including:
  1. Traditional JavaScript Object Model (NS 2 & IE 3)
  2. Extended Traditional JavaScript Object Model (NS 3)
  3. Dynamic HTML Flavored Object Models
     1. a. IE 4
     2. b. NS 4
  4. Traditional Browser Object Model + Standard DOM (NS6 & Explorer 5)



Overview of Core Objects

Object	Description
Window	The object that relates to the current browser window.
Document	An object that contains the various HTML elements and text fragments that make up a document. In the traditional JavaScript object model, the **Document** object relates roughly the HTML **<body>** tag.
Frames[ ]	An array of the frames in the Window contains any. Each frame in turn references another **Window** object that may also contain more frames.
History	An object that contains the current window's history list, namely the collection of the various URLs visited by the user recently.
Location	Contains the current location of the document being viewed in the form of a URL and its constituent pieces.
Navigator	An object that describes the basic characteristics of the browser, notably its type and version.

Document Object

- The **Document** object provides access to page elements such as anchors, form fields, and links as well as page properties such as background and text color.
- Consider
  - document.alinkColor, document.bgColor, document.fgColor, document.URL
  - document.forms[ ], document.links[ ], document.anchors[ ]
- We have also used the methods of the **Document** object quite a bit
  - document.write( ) , document.writeln( ), document.open( ), document.close( )

Object Access by Document Position

- HTML elements exposed via JavaScript are often placed in arrays or collections. The order of insertion into the array is based upon the position in the document.
- For example, the first **<form>** tag would be in document.forms[0], the second in document.forms[1] and so on.
- Within the form we find a collection of elements[ ] with the first **<input>, <select>** or other form field in document.forms[0].elements[0] and so on.
- As arrays we can use the length property to see how many items are in the page.
- The downside of access by position is that if the tag moves the script may break

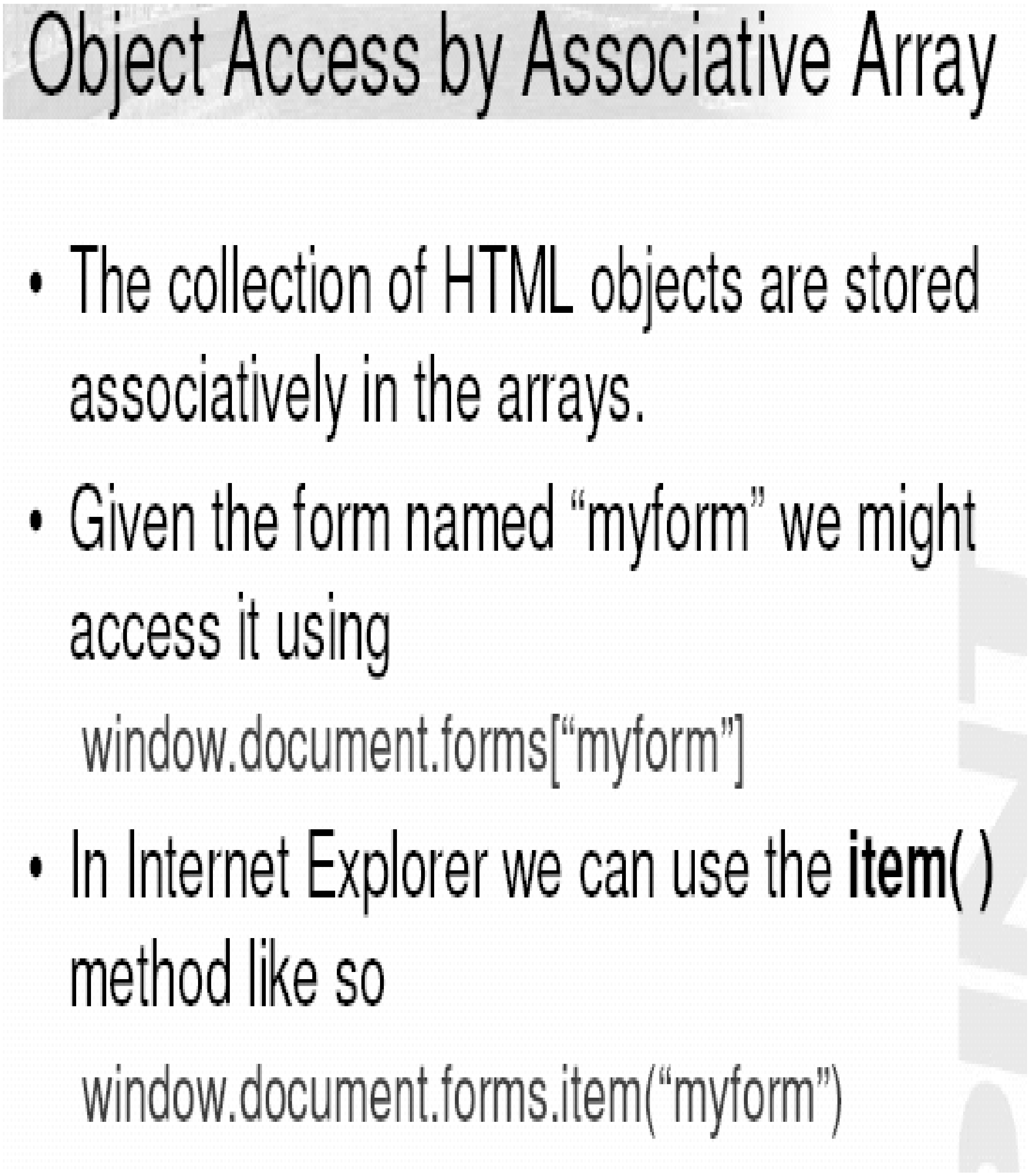
Object Access by Name

- When a tag is named via the **name** attribute (HTML 4.0 - **<a>, <img>,** embedded objects, form elements, and frames) or by **id** attribute (pretty much every tag) it should be scriptable.
- Given

```
<form id="myform" name="myform">
  <input type="text" name="username" id="username">
</form>
```

we can access the form at window.document.myform and the first field as

window.document.myform.username



Object Access by Associative Array

- The collection of HTML objects are stored associatively in the arrays.
- Given the form named “myform” we might access it using

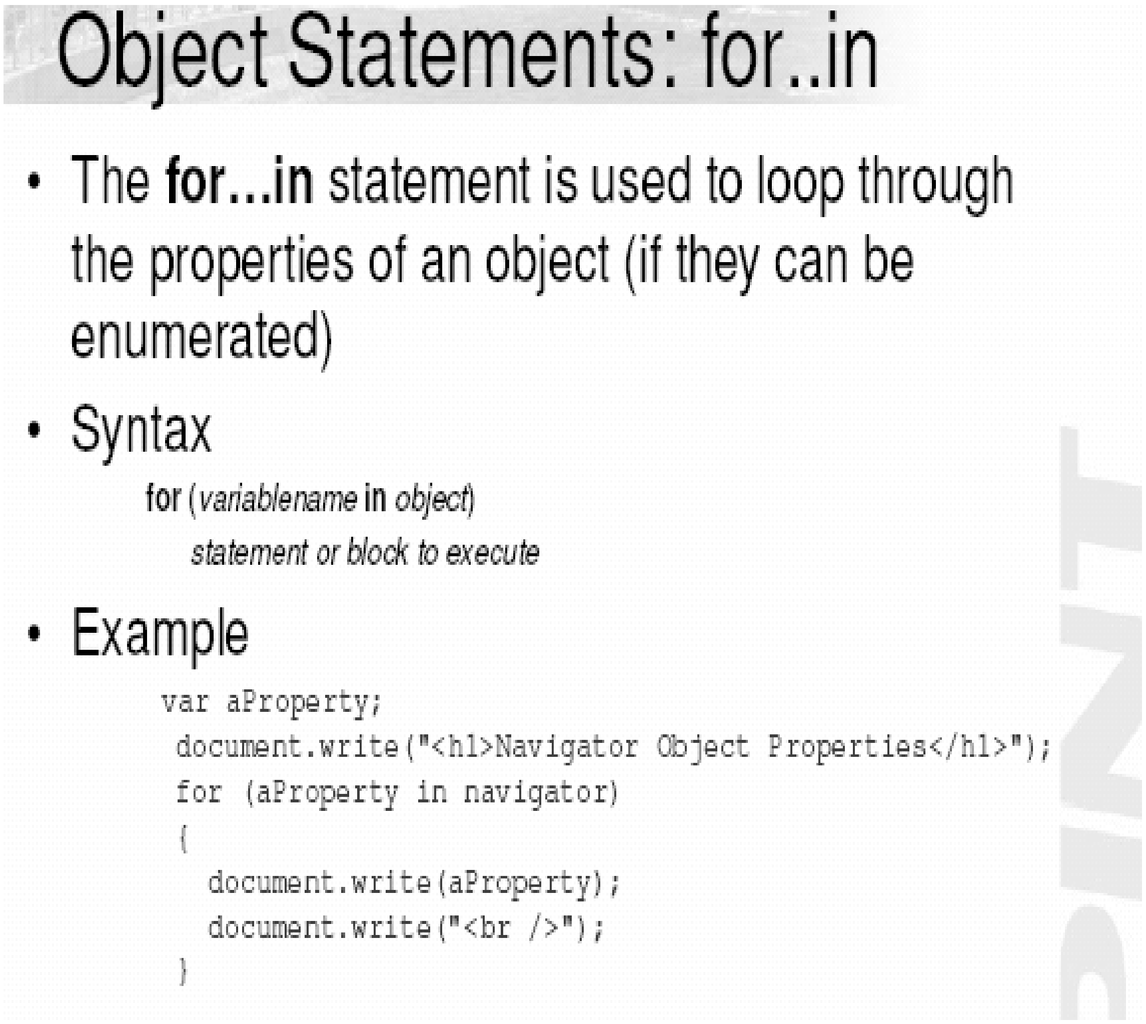
  window.document.forms[“myform”]
- In Internet Explorer we can use the **item( )** method like so

  window.document.forms.item(“myform”)

Object Statement: With

```
document.write(“Hello World”);
document.write(“<br />”);
document.write(“this is another write statement”);
```

```
with (document) {
          write("Hello World");
           write("<br />");
           write("this is another write statement");
}
```



Object Statements: for..in

- The **for...in** statement is used to loop through the properties of an object (if they can be enumerated)
- Syntax

  for (*variablename* in *object*)

  statement or block to execute
- Example

```
var aProperty;
 document.write("<h1>Navigator Object Properties</h1>");
 for (aProperty in navigator)
 {
   document.write(aProperty);
   document.write("<br />");
 }
```

Events

- One of the primary uses of JavaScript is to make Web pages interactive.
  - Responsive to user actions
- JavaScript provides event handlers.
  - Execute segment of code based on events occurring within the application
  - E.g., `onLoad` or `onClick`
- Handlers associated with elements.
- Not all elements support all event handlers.

Events (cont.)

```
<input type="button"
   name="clickme"
   value="Click Here"
   onClick=
      "window.status='Thanks';
     return true;">
```

Events (cont.)

- Event handlers can be categorized into interactive and non-interactive.
- Interactive: Depends on a user action.
  - E.g., `onClick`
- Non-interactive: Non-user event.
  - E.g., `onLoad`

Events (cont.)

- `onAbort`: Image loading is interrupted.
- `onBlur`: Element loses input focus.
- `onChange`: User selects or deselects item.
- `onClick`: User clicks once.
- `onDragDrop`:
- `onError`: Image doesn't load properly.

Events (cont.)

- `onFocus`: Element is given input focus.
- `onKeyPress`:
- `onKeyUp`:
- `onLoad`:
- `onMouseDown`:
- `onMouseOver`:
- `onMouseOut`:
- `onMouseUp`:

Events
Supporting Events

1. Give the target HTML element a `name` attribute.

```
<input type="text"
       name="price"  />
```

2. Give activating HTML element event attribute that calls function.

```
<input type="submit"
    value="Calculate total."
    onClick="calcTotal()"/>
```