Web Technology & Programming

JavaScript: Introduction to Client Side Scripting

LECTURER: KRUNAL PATEL
ICCT ENGG. COLLEGE
NEW VALLABH VIDYANAGAR

Introduction

- JavaScript was originally called LiveScript and was developed by Netscape Communications.

What is JavaScript?

- JavaScript is a Scripting language designed for Web pages.

Why Use JavaScript?

- JavaScript enhances Web pages with dynamic and interactive features.
- JavaScript runs in client software.
- Scripts are programs just like any other programs.

Introduction

What Can JavaScript Do?

- Common JavaScript tasks can replace server-side scripting.
- JavaScript enables form validation, calculations, special graphic and text effects, image swapping, image mapping, clocks, and more.
- Handling User Interaction
  - Doing small calculations
  - Checking for accuracy and appropriateness of data entry from forms
  - Doing small calculations/manipulations of forms input data
  - Search a small database embedded in the downloaded page
  - Save data as cookie so it is there upon visiting the page

Introduction

- Generating Dynamic HTML documents

Examples

- Bookmark lets
- Google Maps
- Google Suggest

Advantages of JavaScript

- Less server interaction
- Immediate feedback to the visitors
- Increased interactivity
- Richer interfaces
- Web surfers don’t need a special plug-in to use your scripts.
- Java Script is relatively secure.

Limitations with JavaScript

- We can not treat JavaScript as a full fledged programming language.

- It lacks the important features like:

  - **Not allow the reading or writing of files.**
  - **It can not be used for Networking**
  - **It doesn't have any multithreading or multiprocessor capabilities**.

JavaScript vs. Java

- Complementary
  - JavaScript
    - Cannot multi-thread, network or do I/O
  - Java
    - Cannot interact with Browser or control content

JavaScript vs. Java

- JavaScript is becoming what Java was originally intended to be
  - Java Applets are meant to be lightweight downloadable programs run within the browser for cross-platform compatibility
  - Java = full
  - JavaScript is actually lightweight and accomplish most of what Applets do with a fraction of the resources

JavaScript Syntax.

- Unlike HTML, JavaScript is case sensitive.
  - HTML is not case sensitive; onClick, ONCLICK, … are HTML
- Dot Syntax is used to combine terms.
  - e.g., **`document.write("Hello World")`**
- Certain characters and terms are reserved.
- JavaScript is simple text

JavaScript Terminology.

- JavaScript programming uses specialized terminology.
- Understanding JavaScript terms is fundamental to understanding the script.
  - Objects, Properties, Methods, Events, Functions, Values, Variables, Expressions, Operators.

JavaScript Structure

- **<script type="text/javascript">**

  and end with

  </script>

- **<script type="text/javascript">**

  <!--

  and end with

  // -->

  </script>

JavaScript Example

```
<html>
  <head>
      <script type="text/javascript">
            document.write("Simple javascript");
      </script>
  </head>
</html>
```

Methods of Using JavaScript.

1. JavaScripts can reside in a separate page.
2. JavaScript can be embedded in HTML documents -- in the **<head>**,or in the **<body>**
3. JavaScript object attributes can be placed in HTML element tags.

   e.g., **<body onLoad="alert('WELCOME')">**

1. Using Separate JavaScript Files.

- Linking can be advantageous if many pages use the same script.
- Use the source element to link to the script file.

```
<script src="myjavascript.js”
    language="JavaScript”
    type="text/javascript">
</script>
```

2. Embedding JavaScript in HTML.

- When specifying a script only the tags **<script>** and **</script>** are essential, but complete specification is recommended:

```
<script language="javascript"
    type="text/javascript">
      <!-- Begin hiding


        // End hiding script-->
</script>
```

Using Comment Tags

- HTML comment tags should bracket any script.
- The **`<!-- script here -->`** tags hide scripts in HTML and prevent scripts from displaying in browsers that do not interpret JavaScript.
- Double slashes **`//`** are the signal characters for a JavaScript single-line comment.

3. Using JavaScript in HTML Tags.

- Event handlers like onMouseover are a perfect example of an easy to add tag script.

```
<a href=”index.html”
   onMouseover="document.logo.src='js2.gif'"
    onMouseout="document.logo.src='js.gif'">
<img src="js.gif" name="logo">
</a>
```

Creating an Alert Message

- The following script in the **<body>** tag uses the **onLoad** event to display an Alert window
- The message is specified within parenthesis.

```
<body onLoad="alert('WELCOME.  Enjoy
your visit.  Your feedback can improve
cyberspace.  Please let me know if you
detect any problems.  Thank you.')">
```

JavaScript Pop-up Boxes

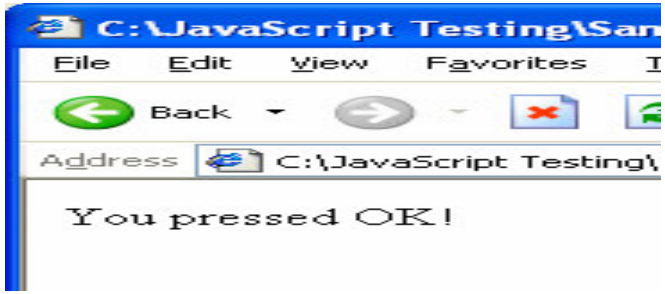
- Alert Box
- Confirm Box
- Prompt Box

Alert Box

```
<html>
<head>
<script language="javascript">
function disp_alert()
{
alert("Hello again!This is how we" + '\n' +
"add line breaks to an alert box!")
}
</script>
</head>
<body>
<input type="button"
onclick="disp_alert()" value="Display
alert box" />
</body>
</html>
```

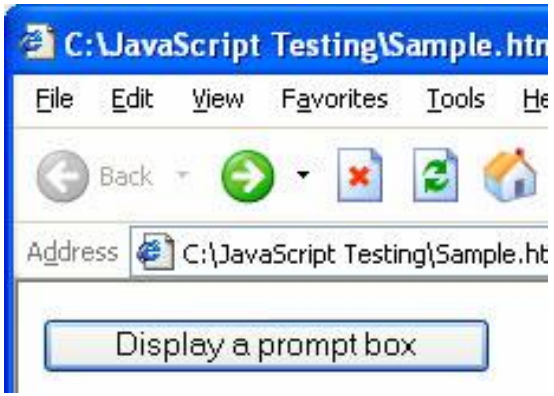
Confirm Box

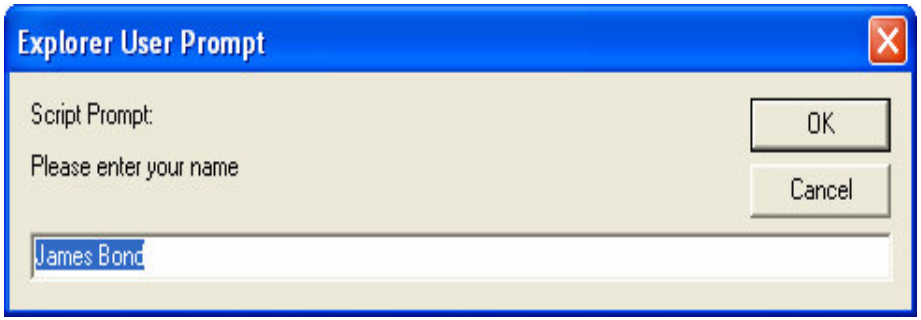
```
<html>
<head>
<script language="javascript">
function disp_confirm()
{
var r=confirm("Press a button")
if (r==true)
{
document.write("You pressed OK!")
}
else
{
document.write("You pressed Cancel!")
}
}
</script>
</head>
<body>
<input type="button" onclick="disp_confirm()"
value="Display a confirm box" />
</body>
</html>
```



Prompt Box

```
<html>
<head>
<script language="JavaScript">
function disp_prompt()
{
var name=prompt("Please enter your name",”@")
if (name!=null && name!="")
{
document.write("Hello " + name + "! How are
you
today?")
}
}
</script>
</head>
<body>
<input type="button" onclick="disp_prompt()"
value="Display a prompt box">
</body>
</html>
```





JavaScript Syntax - Variables and Literals

- Variables contain values and use the equal sign to specify their value.
- Variables are created by declaration using the **`var`** command with or without an initial value state.
  - e.g. **`var month;`**
  - e.g. **`var month = April;`**

JavaScript Syntax - Variables and Literals

- Declaration
  - Explicit: var i = 12;
  - Implicit: i = 12;

- Variable Scope
  - Global
    - Declared outside functions
    - Any variable *implicitly defined*
  - Local
    - *Explicit declarations inside functions*

JavaScript Syntax - Variables and Literals

- Dynamic Typing - Variables can hold any valid type of value:
  - Number … var myInt = 7;
  - Boolean … var myBool = true;
  - Function … [Discussed Later]
  - Object … [Discussed Later]
  - Array … var myArr = new Array();
  - String … var myString = "abc";
  - … and can hold values of different types at different times during execution

JavaScript Syntax - Operators

Key Comparison Operators

>	number on the left must be greater than the number on the right
<	number on the right must be greater than the number on the left
<=	number on the right must be greater than or equal to the number on the left
>=	number on the right must be less than or equal to the number on the left
!=	the numbers or objects or values must not be equal
==	the numbers or objects or values must be equal
!	Logical NOT
||	Logical OR
&&	Logical AND

Datatype Conversion

- 3 + 3 // results in 6
- 3 + “3” // results in “33”
- 3 + 3 + “3” /results in “63”

Converting String to Number

- parseInt(“33”); // results in 33
- parseInt(“33.33); // results in 33
- parseFloat(“33”); // results in 33
- parseFloat(“33.33”); // results in 33.33
- 3 + 3 + parseInt(“3”); // results in 9

Converting Number to String

- “” + 2500 // results in “2500”
- (“” + 2500).length // results in 4

Keywords

ECMA-262 describes a set of *keywords* that ECMAScript supports. These keywords indicate beginnings and/or endings of ECMAScript statements. By rule, keywords are reserved and cannot be used as variable or function names. Here is the complete list of ECMAScript keywords:

```
break       else        new         var
case        finally     return      void
catch       for         switch      while
continue    function    this        with
default     if          throw
delete      in          try
do          instanceof  typeof
```

If you use a keyword as a variable or function name, you will probably be greeted with an error message like this: "Identifier expected."

Reserved Words

ECMAScript also defines a number of *reserved words*. The reserved words are, in a sense, words that are reserved for future use as keywords. Because of this, reserved words cannot be used as variable or function names. The complete list of reserved words in ECMA-262 Edition 3 is as follows:

```
abstract    enum        int         short
boolean     export      interface   static
byte        extends     long        super
char        final       native      synchronized
class       float       package     throws
const       goto        private     transient
debugger    implements  protected   volatile
double      import      public
```

JavaScript Syntax - Operators

Key Assignment Operators

+	adds two numbers or appends two strings. If more than one type of variable is appended, including a string appended to a number or vice-versa, the result will be a string
-	subtracts the second number from the first
/	divides the first number by the second
*	multiplies two numbers
%	Modulus - divide the first number by the second and return the remainder
=	assigns the value on the right to the object on the left
+=	the object on the left = the object on the left + the value on the right this also works when appending strings
-=	the object on the left = the object on the left - the value on the right
++ / --	Increment / Decrement a number

Arrays

- Creating Arrays

```
var a = new Array(); // empty array
var b = new Array("dog", 3, 8.4);
var c = new Array(10); // array of size 10
var d = [2, 5, 'a', 'b'];
```

- Assigning values to Arrays

```
c[15] = "hello";
c.push("hello");
```

JavaScript Syntax – Control and Looping

- JavaScript supports conditional statements which are used to perform different actions based on different conditions.

- JavaScript supports following forms of **if..else** statement:
    1. if statement
    2. if…else statement
    3. if…else if… statement.

JavaScript Syntax – Control and Looping

- **Switch Case Stmt**

  The basic syntax of the **switch** statement is to give an expression to evaluate and several different statements to execute based on the value of the expression. The interpreter checks each **case** against the value of the expression until a match is found. If nothing matches, a **default** condition will be used.

```
switch (expression)
{
        case condition 1: statement(s) break;
        case condition 2: statement(s) break;
        … case condition n: statement(s) break;
        default: statement(s)
}
```

JavaScript Syntax – Control and Looping

- A **loop** is a set of instructions that is executed repeatedly
- The loop uses a **counter** to track the number of times the command block has been run
- Loops execute a block of code a specified number of times, or while a specified condition is true.

The for Loop

```
var i=0;
for (i=0;i<=5;i++)
{
        document.write("The number is " + i);
        document.write("<br />");
}
```

JavaScript Syntax – Control and Looping

The while Loop

```
while (variable<=endvalue)
 {
 code to be executed
 }
```

The do..While loop

```
do
 {
 code to be executed
 }
while (variable<=endvalue);
```

The break / continue Statement

-
```
for (i=0;i<=10;i++)
 {
 if (i==3)
  {
  break ;    or    ( Continue)
  }
 document.write("The
 number is " + i);
 document.write("<br />");
 }
```

Developing More Advanced Scripts

Objects

- JavaScript is an Object based Programming language
- A JavaScript object has properties and methods

Example: String JavaScript object has length property and toUpperCase() method

```
<script type="text/javascript">
        var txt="Hello World!“
        document.write(txt.length)
        document.write(txt.toUpperCase())
</script>
```

- String , Date , Array , Boolean and Math are Built in javascript objects.

Developing More Advanced Scripts

Properties

- Properties are object attributes.
- It determines the functionality of the object.
- Object properties are defined by using the object's name, a period, and the property name.

e.g., background color is expressed by: document.bgcolor .

- document is the object.
- bgcolor is the property.

Developing More Advanced Scripts

Methods

- Methods are actions applied to particular objects.
- Methods are what objects can do.

e.g., document.write("Hello World")

- document is the object.
- write is the method.

User-Defined Objects

The new Operator:

- The new operator is used to create an instance of an object.
- To create an object, the new operator is followed by the constructor method.
- In the following example, the constructor methods are Object(), Array(), and Date().
- These constructors are built-in JavaScript functions.

For Example:

```
var employee = new Object();
var books = new Array("C++", "Perl", "Java");
var day = new Date("August 15, 1947");
```

JavaScript Object

- A constructor is a function that creates and initializes an object.

Adding Constructor

- A constructor is a pre-defined method that will initialize your object.
- A constructor is invoked through the new operator.
- Properties are assigned using this keyword.

Example

```
function person (firstname, age)
{
this.firstname=firstname;
this.age=age;
}
```

```
Var person1 = new
person("smith",32);

Document.write(person1.name +
person1.age)
```

Developing More Advanced Scripts

Object Function

Object Functions

Then to call the function on the object

```
var someGuy = new Person("Shawn", 28);
someGuy.displayMe();

var someOtherGuy = new Person("Tim", 18);
someOtherGuy.displayMe();
```

Developing More Advanced Scripts

Object Function

Object Functions

Use a function to construct an object

```
function car(make, model, year)
{
        this.make = make;
        this.model = model;
        this.year = year;
}
```

Developing More Advanced Scripts

Many other pre-defined object types

- **String:** manipulation methods
- **Math:** trig, log, random numbers
- **Date:** date conversions
- **RegExp:** regular expressions
- **Number:** limits, conversion to string

Predefined Objects

- JavaScript also includes some objects that are automatically created for you (always available).
  - document
  - navigator
  - screen
  - window

The document object

- Many objects descend from the document object forming a large sub tree known as the document object model
- Many attributes of the current document are available via the document object:

The document object

- Document object represents the HTML displayed in the window

```
document.bgcolor=red
document.linkcolor=yellow
document.write ("<h2> Hello World </h2>);
```

- you can access any form information in a document by using the form[] array.

The document object

- Example

```
<form name="userdetails">
<input type="text" name="fname"/>
<input type="text" name="lname"/>
<input type="submit" name="submit"/>
</form>
document.forms[0]
```

- it can also be referred by **document.userdetails** any element within it can accessed by

document.userdetails.fname

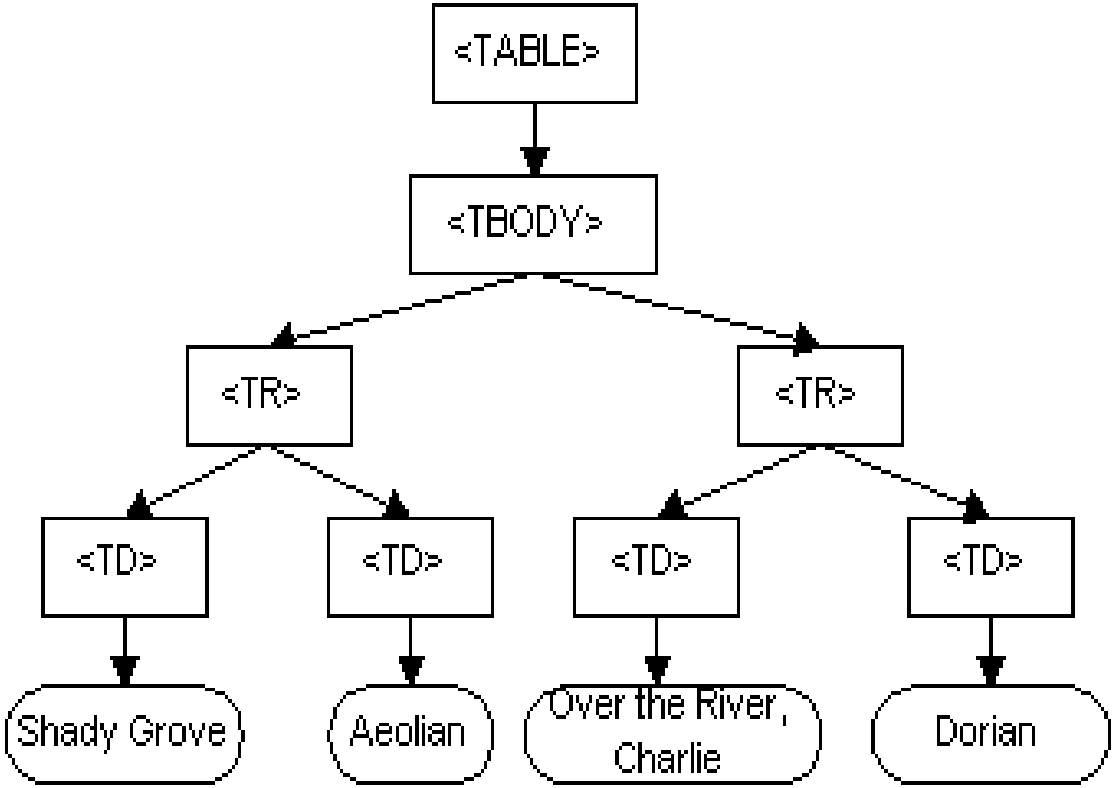
Document Object Model (DOM)

- An object-based, language-neutral API for XML and HTML documents.
- It defines the logical structure of documents and the way a document is accessed and manipulated
- It allows programs and scripts to build documents, navigate their structure, add, modify or delete elements and content
- Provides a foundation for developing querying, filtering, transformation, rendering etc. applications on top of DOM implementations

Document Object Model (DOM)

```
<TABLE>
<TBODY>
<TR>
<TD>Shady Grove</TD>
<TD>Aeolian</TD>
</TR>
<TR>
<TD>Over the River, Charlie</TD>
<TD>Dorian</TD>
</TR>
</TBODY>
</TABLE>
```

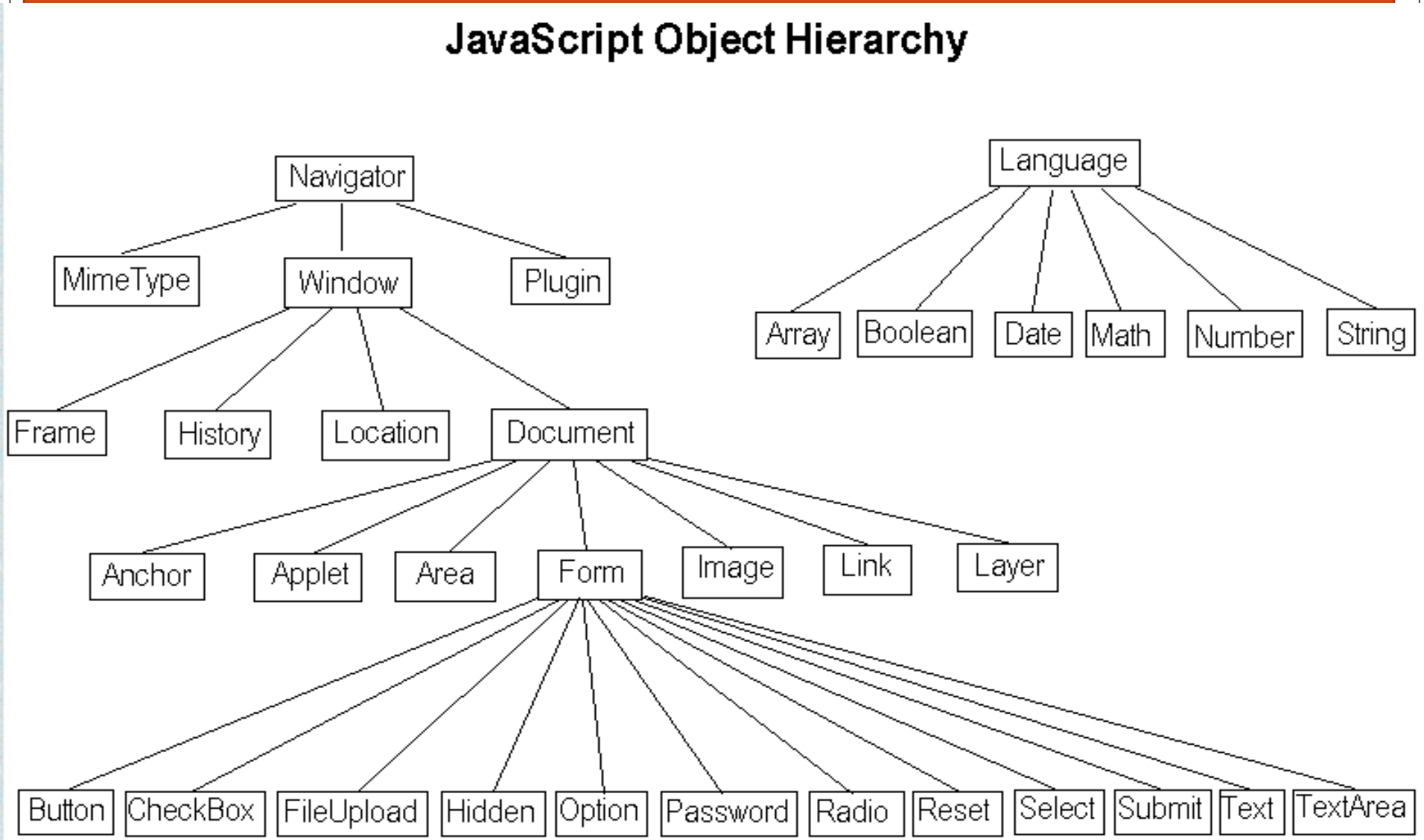
Document Object Model (DOM)



Document Object Model (DOM)

- JavaScript access to the elements of an HTML document.
- An object hierarchy
- Provides access to elements via:
  - ID (**`ID`** attribute of HTML tags)
  - Order of elements in document
  - Order of element in collection of similar elements.

Document Object Model (DOM)



Document Object Model (DOM)

Browser Objects

- Window
- Navigator
- Screen
- History
- Location

Document Object Model (DOM)

Window object

- window object is primary point from which other objects come.
- 'window' object is JavaScript representation of a browser window.
- window object represents the window or frame that displays the document and it is the global object in client side programming
  - **closed** - A boolean value that indicates whether the window is closed.
  - **defaultStatus** - This is the default message that is loaded into the status bar when the window loads.

```
window.defaultStatus = "This is the status bar";
```

Document Object Model (DOM)

Window object

Window

Self , Parent , top
frames[ ]
navigator
location
history
document
screen

- **self :** current window **Parent :** parent window **top :** for a frame top level window that contains it.
- **frames[ ]** : Array of window objects , one for each frame
- **navigator :** Represents the browser. Read-only!

Document Object Model (DOM)

Window object

Attributes include: appName ,appVersion, platform

navigator.appName == "Microsoft Internet Explorer"

- **location :** a reference to location object which holds the current URL
- **history** : history object which holds the history of the user's visit
- **screen :** it will give information about size of user's display and color depth.

  attributes : width , height , availwidth , availheight and colorDepth

  availWidth and availHeight specifies the space actually available excluding task bar and other screen borders

Document Object Model (DOM)

Window object

Window – Built-in Functions

alert("message")

window.close()

confirm("message")

focus()

open("URLname","Windowname",["options"])

open(“http://google.com”, “My Google”,

“toolbar=no,alwaysRaised=yes”);

Location Object

- The location object contains information about the current URL.
- The location object is part of the window object and is accessed through the **window. Location** property.
- The location object contains all the information on the location that the window is currently displayed and all the details on that location like port, the protocol.

Properties

- **Href** - The href property is the entire URL of the current page. Using this property, you can specify the location of the other URL and its equivalent to typing URL in the address bar
- **Host** - The hostname is the name of the host machine on which the current URL is located
- **Port** - If specific port is specified in the URL, its value is located in the port property

Document Object Model (DOM)

Location Object Properties

Property	Description
hash	Returns the anchor portion of a URL
host	Returns the hostname and port of a URL
hostname	Returns the hostname of a URL
href	Returns the entire URL
pathname	Returns the path name of a URL
port	Returns the port number the server uses for a URL
protocol	Returns the protocol of a URL
search	Returns the query portion of a URL

Location Object Methods

Method	Description
assign()	Loads a new document
reload()	Reloads the current document
replace()	Replaces the current document with a new one

Document Object Model (DOM)

Example

Return the query portion of a URL.

Assume that the current URL is

http://www.example.com/submit.htm?email=someone@example.com:

The output of the code above will be:

```
<script type="text/javascript">
        document.write(location.search);
</script>
```

?email=someone@example.com

Navigator Object

- The navigator object enables to access general information about the browser program. – What the browser does and does not support.The Navigator object is automatically created by the JavaScript runtime engine and contains information about the client browser.
- Properties
  - appName – returns name of the browser that is processing the script
  - **appVersion** – returns version number of the browser
  - **userAgent** -The userAgent property supplies the user agent, which is the exact string that is sent via HTTP as user-agent header when communicating with a web server: browser Language - Returns the current browser language
  - **cookieEnabled** - Returns a Boolean value that specifies whether cookies are enabled in the browser
  - **cpuClass** - Returns the CPU class of the browser's system
  - **onLine** - Returns a Boolean value that specifies whether the system is in offline mode
  - **platform** - Returns the operating system platform
  - **systemLanguage** - Returns the default language used by the OS
  - **userAgent** - Returns the value of the user-agent header sent by the client to the server
  - **userLanguage** - Returns the OS' natural language setting

Document Object Model (DOM)

Navigator Object

The navigator object contains information about the browser.

```
<script type="text/javascript">

document.write("Name: " + navigator.appName);
document.write("CodeName: " + navigator.appCodeName);
document.write("Cookies enabled: " + navigator.cookieEnabled);
document.write("Platform: " + navigator.platform);
document.write("User-agent header sent: " + navigator.userAgent);
document.write("Java enabled: " + navigator.javaEnabled());


</script>
```

Document Object Model (DOM)

History Object

- The History object is automatically created by the JavaScript runtime engine and consists of an array of URLs.
- These URLs are the URLs the user has visited within a browser window.
- The History object is part of the Window object and is accessed through the window.history property.
- history.length – The length property specified how many URLs are contained in the current history object.

Document Object Model (DOM)

Methods of history object

- history.forward() – Loads the next URL in the history list
- history.back() - Loads the previous URL in the history list
- history.go(URL or no.) - Loads a specific page in the history list

```
<html>
<body>
<script language=”JavaScript”>
document.write(history.length);
</script>
</body>
</html>
```

Events

- An event 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)" >
```

Handle Key Events

- Like the user interface events, key events fire in a predictable sequence. There are three main key events in HTML.

Event Name	Event Handler Name
keypress	onkeypress
keydown	onkeydown
keyup	onkeyup

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

Form

- JavaScript provides access to the forms within a HTML document through the **Form** object , which is a child of the **Document** object.
- As with all **document** objects, the properties and methods of this object correspond to the various features and attributes of the HTML <**form**>.

Form

- Processing of submitted information on the client side is advantage in terms of resources – by not sending the data over to the server
- JavaScript can be used to validate data in HTML forms before sending off the content to a server.

Form data that typically are checked by a JavaScript could be:

has the user left required fields empty?

has the user entered a valid e-mail address?

has the user entered a valid date?

has the user entered text in a numeric field?

Properties of the Form object

Property	Description
action	Holds the value of the **action** attribute indicating the URL to send the form data to when the user submits.
elements[]	Array of form fields objects representing the form field elements enclosed by this <**form**>.
length	The number of form fields within this <**form**> tag. Should be the same as **elements.length**.
method	The value of the **method** attribute of this <**form**> tag. Should be either **GET** or **POST**.
name	The name of the <**form**> as defined by its **name** attribute. You probably should also set the **id** attribute to hold the same value.
target	The name of the frame in which to display the page resulting from form submission.

Form Methods

Forms also have two form-specific methods.

- The **reset()** method clears the form's fields, similar to clicking a button defined by **<input type="reset" />**
- The **submit()** method triggers the submission of the form similar to clicking the button defined by **<input type="submit" />**.
- In addition to the ability to trigger form reset and submission, we often want to react to these events as well
- so the **<form>** tag supports the corresponding **onreset** and **onsubmit** event handler attributes.
- As with all events, handler scripts can return a value of **false** to cancel the reset or submit.
- Returning **true** (or not returning a value) permits the event to occur normally.

Accessing Forms

- Before manipulation of form fields, we need to have the capable of accessing a **Form** fields.
- Forms can be accessed in three ways:
  - by number through **document.forms[].**
  - by name through **document.forms[].**
  - by the regular element retrieval mechanisms (e.g., **document.*formname***)

- <form name="customerform" id="customerform" >
  <input type="text" name="firstname=" id="firstname=" />
  <input type="text" name="lastname=" id="lastname=" /> …more fields… </form>

- we might use **window.document.forms[0]** (assuming it's the first form in the page), **window.document.forms['customerform']**, or **window.document.customerform**

Accessing Form Fields

- Just as the **Document** contains a collection of **<form>** tags, each form contains a collection of form fields that can be accessed through the **elements[]** collection.
- **window.document.customerform.elements[0]** refers to the first field.
- Similarly, we could access the fields by name,

window.document.customerform.elements["firstname"] or

window.document .customerform.firstname

Accessing Form Fields

- Just as the **Document** contains a collection of **<form>** tags, each form contains a collection of form fields that can be accessed through the **elements[]** collection.
- **window.document.customerform.elements[0]** refers to the first field.
- Similarly, we could access the fields by name,

window.document.customerform.elements["firstname"] or

window.document .customerform.firstname