Lecture 9

JavaScript

Page Redirection

- To redirect your site visitors to a new page, you just need to add a line in your head section.

```
<script type="text/javascript">
window.location="http://www.newlocation.com
";
</script>
```

Page Printing

- JavaScript helps you to implement this functionality using print function of ***window*** object.
- The JavaScript print function **window.print()** will print the current web page when executed. You can call this function directly using ***onclick*** event.

```
<form>
<input type="button" value="Print" onclick="window.print( )" />
</form>
```

Loops

- **For**
- **While**
- **Do while**

Exceptions

- **Exceptions can be handled with the common try/catch/finally block structure.**
  - The try block must be followed by either exactly one catch block or one finally block (or one of both).
  - When an exception occurs in the catch block, the exception is placed in **e** and the catch block is executed. The finally block executes unconditionally after try/catch

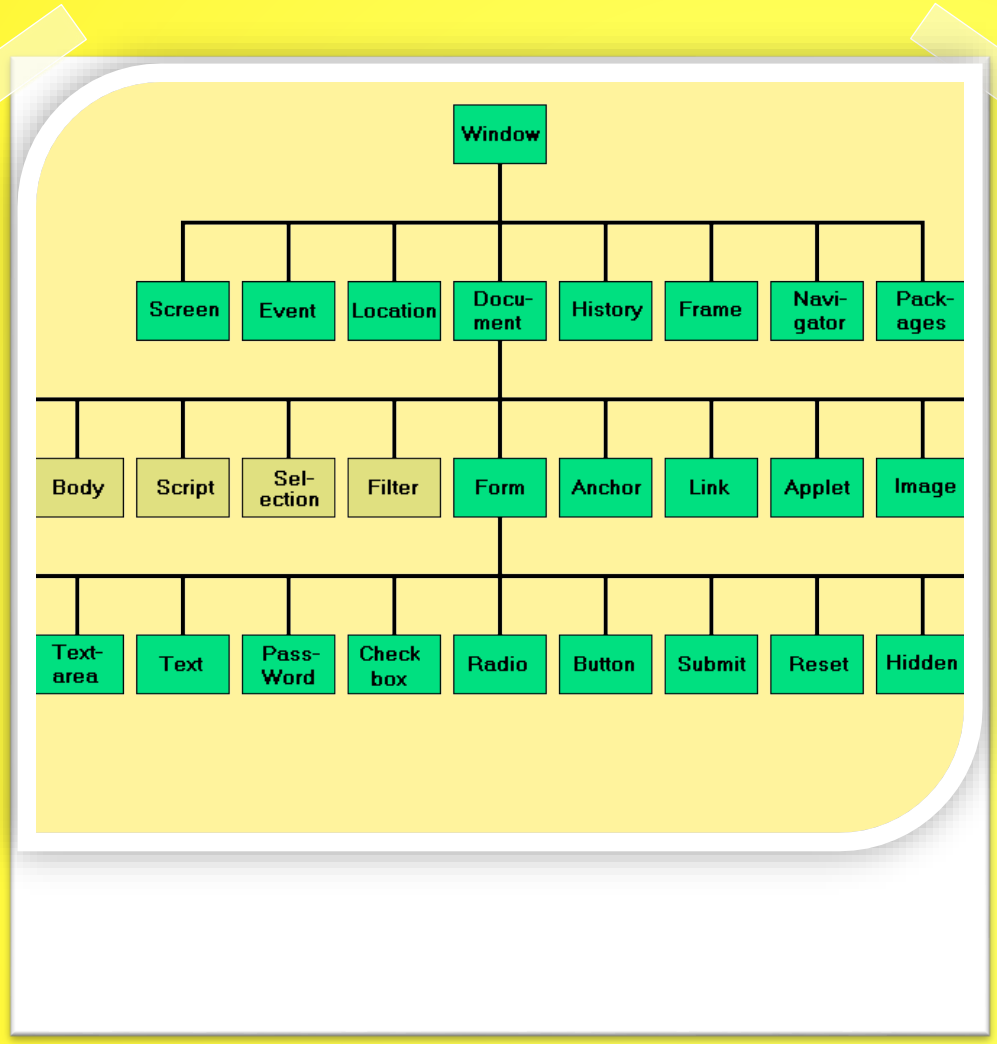
```
<script type="text/javascript">
try {
   statementsToTry
} catch ( e ) {
      catchStatements
} finally {
      finallyStatements
}
</script>
```

Numeric Methods

Method	Description
constructor()	Returns the function that created this object's instance. By default this is the Number object.
toExponential()	Forces a number to display in exponential notation, even if the number is in the range in which JavaScript normally uses standard notation.
toFixed()	Formats a number with a specific number of digits to the right of the decimal.
toLocaleString()	Returns a string value version of the current number in a format that may vary according to a browser's locale settings.
toPrecision()	Defines how many total digits (including digits to the left and right of the decimal) to display of a number.
toString()	Returns the string representation of the number's value.
valueOf()	Returns the number's value.

String Methods

Method	Description
charAt()	Returns the character at the specified index.
charCodeAt()	Returns a number indicating the Unicode value of the character at the given index.
concat()	Combines the text of two strings and returns a new string.
indexOf()	Returns the index within the calling String object of the first occurrence of the specified value, or -1 if not found.
lastIndexOf()	Returns the index within the calling String object of the last occurrence of the specified value, or -1 if not found.
length()	Returns the length of the string.
match()	Used to match a regular expression against a string.
replace()	Used to find a match between a regular expression and a string, and to replace the matched substring with a new substring.
search()	Executes the search for a match between a regular expression and a specified string.
slice()	Extracts a section of a string and returns a new string.
split()	Splits a String object into an array of strings by separating the string into substrings.
substr()	Returns the characters in a string beginning at the specified location through the specified number of characters.
substring()	Returns the characters in a string between two indexes into the string.
toLocaleLowerCase()	The characters within a string are converted to lower case while respecting the current locale.
toLocaleUpperCase()	The characters within a string are converted to upper case while respecting the current locale.
toLowerCase()	Returns the calling string value converted to lower case.
toString()	Returns a string representing the specified object.
toUpperCase()	Returns the calling string value converted to uppercase.
valueOf()	Returns the primitive value of the specified object.



Document Object Model (DOM)

Document Object Model (DOM)

- Every web page resides inside a browser window which can be considered as an object.
- A Document object represents the HTML document that is displayed in that window.
- The Document object has various properties that refer to other objects which allow access to and modification of document content.
- The way that document content is accessed and modified is called the Document Object Model, or DOM.
- The Objects are organized in a hierarchy.
- This hierarchical structure applies to the organization of objects in a Web document.

Window, Document & Form Objects

- **An object is a set of variables, functions, etc., that are in some way related. They are grouped together and given a name. Objects may have:**
- **Properties**

➲A variable (numeric, string or Boolean) associated with an object. Most properties can be changed by the user.

➲ Example: the title of a document

- **Methods**

➲ Functions associated with an object. Can be called by the user.

➲Example: the alert() method

- **Events**

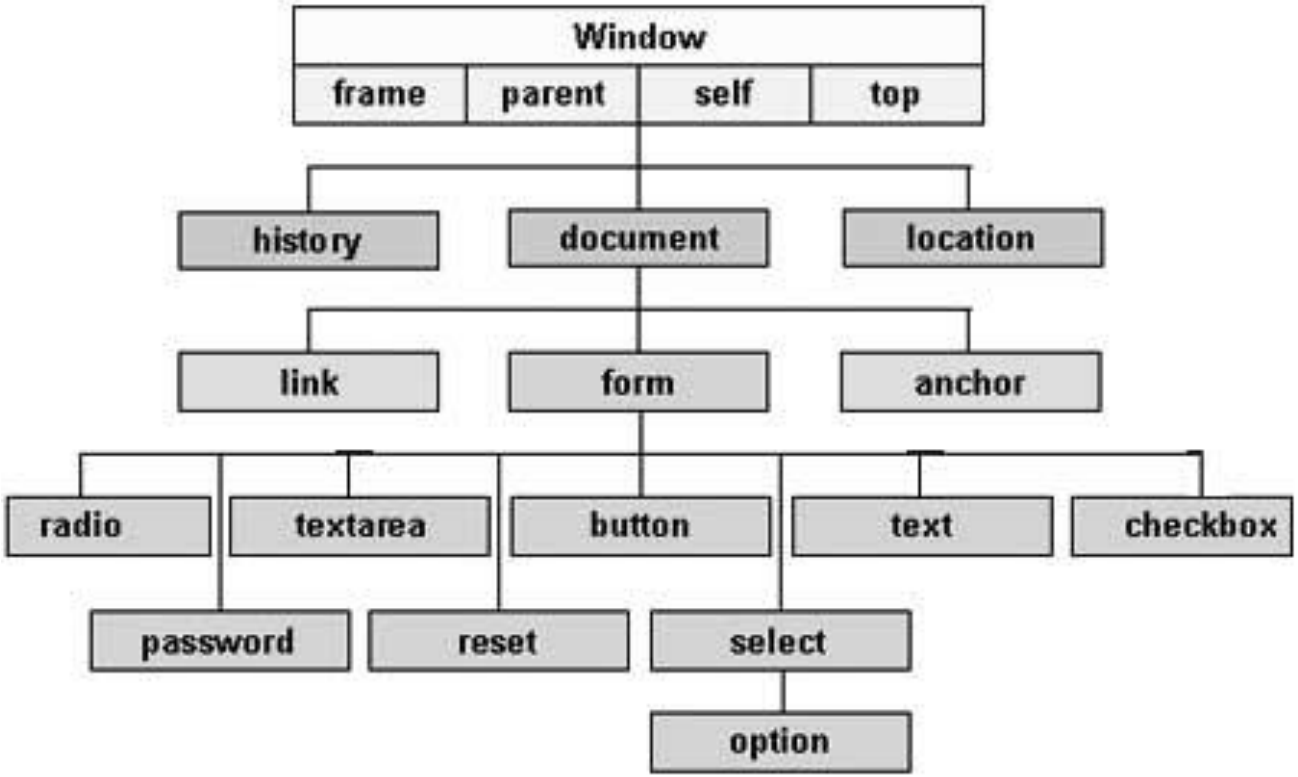
➲Notification that a particular event has occurred. Can be used by the programmer to trigger responses.

➲Example: the onClick() event.

Different Objects of DOM

- **Window object:**
  - Top of the hierarchy. It is the outmost element of the object hierarchy.
- **Document object:**
  - Each HTML document that gets loaded into a window becomes a document object. The document contains the content of the page.
- **Form object:**
  - Everything enclosed in the <form>...</form> tags sets the form object.
- **Form control elements:**
  - The form object contains all the elements defined for that object such as text fields, buttons, radio buttons, and checkboxes.

DOM Diagram


Window
frame
parent
self
top
history
document
location
link
form
anchor
radio
textarea
button
text
checkbox
password
reset
select
option

Window Methods

```
<script type="text/javascript">
alert("Warning Message");
</script>
```

- **Alert Box**
  - To show warning message
- **Prompt Box**
  - input from user

```
var retVal = prompt("Enter your name”);
  alert("You have entered : " + retVal );
```

- **Confirmation Box**

```
<script type="text/javascript">
  var retVal = confirm("Do you want to continue ?");
  if( retVal == true ){
     alert("User wants to continue!");
          return true;
  }else{
     alert("User does not want to continue!");
          return false;
  }
</script>
```

Window Events

- **onLoad()**
  - Message sent each time a document is loaded into a window. Can be used to trigger actions (e.g., calling a function). Usually placed within the <body> tag, for example:
    - **<body onLoad="displayWelcome()">**
  - would cause the function displayWelcome() to execute automatically every time the document is loaded or refreshed.

- **onUnload()**
  - Message sent each time a document is closed or replaced with another document. Can be used to trigger actions (e.g., calling a function). Usually placed within the <body> tag, for example:
    - **<body onUnload="displayFarewell()">**
  - would cause the function displayFarewell() to execute automatically every time the document is closed or refreshed.

Document Object Methods

write() Allows a string of text to be written to the document. Can be used to generate new HTML code in response to user actions.

For example:

```
document.write("<h1>Hello</h1> ");
document.write("<p>Welcome to the new page</p>");
```

Form Objects

- **name**
  - The name of the form, as defined in the HTML <form> tag when the form is created, for example:
    - **<form name="myForm">**
  - This property can be accessed using JavaScript.
    - For example, this paragraph is part of a form that contains the example buttons.
    - It is the third form in the document (the others contain the buttons for the Window and Document object examples). To obtain the name of this form, we could use the following code:
    - **alert(document.forms[2].name);**

Form Objects

- **Method**
  - The method property can be set either to POST or GET.
    - **<form method="POST">**
  - We can access method of form by :
    - **alert(document.forms[2].method);**
- **Length**
  - The number of elements (text-boxes, buttons, etc.) in the form. For example:
  - **alert(document.forms[2].length);**

Form Method & Events

- **submit()**
  - Submits the form data to the destination specified in the action attribute using the method specified in the method attribute.
- **onSubmit()**
  - Message sent each time a form is submitted. Can be used to trigger actions (e.g., calling a function). Usually placed within the <form> tags, for example:
    - **<form onSubmit="displayFarewell()">**
  - Would cause the function displayFarewell() to execute automatically every time the form is submitted.

Text Boxes & Text Areas- Properties

- **Name**
  - The name property of a text-box or other form element can be accessed using JavaScript in the manner shown under the section on document.length, above.
    - `<input type=text name="textBox1">`
- **Value**
  - Getting the value of written in text field.
    - `alert(document.forms[2].textBox1.value);`

Text Boxes & Text Areas- Events

- **onFocus()**
  - ➲Event signal generated when a user **clicks in a text-box** or text-area.
    - **<input type=text name="textBox2“ onFocus="alertOnFocus()">**

- **onBlur()**
  - ➲Event signal generated when a user **clicks outside a text-box** or text-area having previously clicked inside it.
    - **<input type=text name="textBox3" onBlur="alertOnBlur()">**
    -

Buttons, Radio-buttons & Checkboxes- Properties

```
if (document.forms[2].checkbox1.checked == true)
{
    alert("Checked");
}
else
{
    alert("Not checked");
};
```

- **name**
- **Value**
  - We can change the value of the button, and hence it's label,
    - **document.forms[2].button1.value = "New value, new label";**
- **Checked**
  - This property - which is used with radio-buttons and check-boxes but not standard buttons - indicates whether or not the button has been selected by the user.

Buttons, Radio-buttons & Checkboxes- Methods & Events

- **focus()-**
  - Give the button focus (i.e., make it the default button that will be activated if the return key is pressed).
  - **document.forms[2].button2.focus();**
- **blur()**
  - Remove focus
- **click()**
  - Clicking on this button will have the same effect as clicking directly on the button labelled 'Hello‘

Events

Onclick()

onFocus()

onBlur()

Select Object

- Selection-boxes behave in a very similar fashion to radio-buttons: they present several options, of which only one can be selected at a time.
- They also have a similar set of properties, methods and events.
- The principal difference from a programming perspective is that selection-boxes don't have a checked property.
- Instead, to find out which option has been selected, you must use the **SelectedIndex** property

- **selectedIndex**
  - Returns an integer indicating which of a group of options has been selected by the user.
    - **alert(document.forms[2].selectBox1.selectedIndex);**