Unit 4

JAVASCRIPT

2CP08: WEB TECHNOLOGIES3

K J SHARMA

Introduction

- For a web page, HTML supplies document content and structure while CSS provides presentation styling
- In addition, client-side scripts can control browser actions associated with a web page.
- Client-side scripts are almost written in Javascript language to control browser's actions.
- Client-side scripting can make web pages more dynamic and more responsive.

Introduction to JavaScript

- JavaScript was originally called LiveScript and was developed by Netscape Communications.
- JavaScript is a scripting language. A scripting language is a light weight programming language.
- JavaScript is embedded in Web pages and interpreted by the browser.
- A JavaScript was designed to add interactive to HTML pages.

Introduction to JavaScript

- Client-side Programs were developed to run programs and scripts on the client side of a Web browser
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- HTML and CSS concentrate on a static rendering of a page. Things do not change on the page at run time.
- For that we use scripting languages which allows content to change dynamically.
- JavaScript is Case Sensitive.

Comparing Java and JavaScript

- Java and JavaScript are two completely different languages in both concept and design
- Java (developed by Sun Microsystems) is a powerful and much more complex programming language.
- Java is a **compiled language**
- JavaScript is a subset of Java
- JavaScript can put dynamic text into an HTML page
- JavaScript is an **interpreted language**

Comparing Java and JavaScript

Java	JavaScript
A compiled language	An interpreted language
Requires the JDK (Java Developer's Kit) to create the applet	Requires a text editor
Requires a Java virtual machine or interpreter to run the applet	Requires a browser that can interpret JavaScript code
Applet files are distinct from the HTML and XHTML code	JavaScript programs are integrated and can be placed within HTML and XHTML code
Source code is hidden from the user	Source code is made accessible to the user
Powerful, requires programming knowledge and experience	Simpler, requiring less programming knowledge and experience
Secure: programs cannot write content to the hard disk	Secure: programs cannot write content to the hard disk, but there are more security holes than in Java
Programs run on the client side	Programs run on the client side

Task performed by client-side scripts

- Checking **correctness** of user input
- **Monitoring** user events and **specifying reactions**
- **Replacing** and **updating** parts of a page
- Changing the **style** and **position** of displayed elements **dynamically**
- **Modifying** a page in **response** to **events**
- Getting browser **information**
- Making the web page **different** depending on the browser and browsers features

JavaScript Types:

1. **Client Side JavaScript**
2. **Server Side JavaScript**

- **Client Side Scripting** generally refers to the class of computer programs on the web that are **executed at client side** by the user's web browser, instead of server-side (on the web server).

JavaScript Types:

Advantages of ClientSide Scripting

- The Web browser uses it's own resources, and erase the burden on the server.
- Execute quickly because they do not require a trip to the server.
- Can give developers more control over the look and behavior of their Web widgets.

Disadvantages of ClientSide Scripting

- Code is usually visible
- Code is Probably modifiable
- Local files and database can't be accessed

JavaScript Types:

- **Server Side JavaScript**

  Normally when a browser requests an HTML file, the server returns the file , but if the file contains a server side script, the script inside the HTML file is executed by the server before the file is returned to the browser as a plain HTML.

- **What can server scripts do?**
- Respond to user queries of data submitted from HTML forms
- Access any data or databases and return the result for individual users.
- Provide security since your server code cannot be viewed from the browser.

JavaScript Types:

Client Side vs. Server Side

When a client (your computer) makes a request for a web page that information is processed by the web server. If the request is a server side script (e.g. Perl or PHP) before the information is returned to the client the script is executed on the server and the results of the script is returned to the client.

Once the client recieves the returned information from the server if it contains a client side script (e.g. JavaScript) your computer browser executes that script before displaying the web page.

ComputerHope.com

Inserting JavaScript into a Web Page File

- In HTML, JavaScript code is inserted between <script> and </script> tags.
- A JavaScript program can either be placed directly in a Web page file or saved in an external text file
- Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.
- Within head section:

  Use the <script> tag (type attribute to define the scripting language)

```
<html>
<head>
        <script type="text/javascript">
                <!--code goes here--!>
        </script>
</head>
</html>
```

Inserting JavaScript into a Web Page File

- Within body section:

  Use the <script> tag

```
<html>
 <head>
 </head>
      <body>
           <script type="text/javascript">
                <!--code goes here--!>
           </script>
      </body>
 </html>
```

Inserting JavaScript into a Web Page File

- Scripts can also be placed in external files
- External scripts are practical when the same code is used in many different web pages.
- JavaScript files have the file extension .js.
- To use an external script, put the name of the script file in the src (source) attribute of a <script> tag:

```
<script src="myScript.js"></script>
```

JavaScript Output

- An **object-oriented** programming language writes the output by manipulating tasks.
- JavaScript can "display" data in different ways:
  - Writing into the HTML output using **document.write().** - Inside Body
  - Writing into an HTML element, using **innerHTML**
  - Writing into an alert box, using **window.alert().**
  - Writing into the browser console, using **console.log().**

JavaScript Output

Using document.write()

- To write text on Web page, use following JavaScript command:

  document.write("*text*");

  Where *text* is the content to be written to the page.

JavaScript Output

Using innerHTML

- To access an HTML element, JavaScript can use the document.getElementById(id) method.
- The id attribute defines the HTML element. The innerHTML property defines the HTML content

JavaScript Output

Using window.alert()

- You can use an alert box to display data:

```
<script>
    window.alert(5 + 6);
</script>
```

JavaScript Output

Using console.log()

- For debugging purposes, you can use the console.log() method to display data.

```
<script>
    console.log(5 + 6);
</script>
```

Working with Variables and Data

- A **variable** is a named item in a program that stores information
- A variable declared within a JavaScript function becomes **LOCAL** and can only be accessed within that function. (the variable has local scope).
- Local variables are destroyed when you exit the function.
- Variables declared outside a function become **GLOBAL**, and all scripts and functions on the web page can access it.
- Global variables are destroyed when you close the page.

Working with Variables and Data

The general rules for constructing names for variables (unique identifiers) are:

- Names can contain letters, digits, underscores, and dollar signs.
- Names must begin with a letter
- Names can also begin with $ and _
- Names are case sensitive (y and Y are different variables)
- Reserved words (like JavaScript keywords) cannot be used as names

Working with Variables and Data

- **A variable can contain several types of value:**
  - **Numeric or floating variable-** any number, such as 13, 22.5, etc
  - **Boolean variable-** accepts only true and false values
  - **Null variable-** has no value at all
  - **String variable-** any group of text characters, such as "Hello" or "Happy Holidays!"
    - Must be enclosed within either double or single quotations
  - **Function-** a function name
  - **Object-** an object

Declaring a JavaScript Variable

- You can declare variables with any of the following JavaScript commands:
- **var *variable*;**
- **var *variable* = *value*;**
- ***variable* = *value*;**

  Where *variable* is the name of the variable and *value* is the initial value of the variable. The first command creates the variable without assigning it a value; the second and third commands both create the variable and assign it a value.

Working with Operators

- **Operators** are elements that perform actions within expressions
- JavaScript supports the following types of operators.
  - Arithmetic Operators
  - Comparison Operators
  - Logical (or Relational) Operators
  - Assignment Operators
  - Conditional (or ternary) Operators

Working with Operators

Arithmetic Operators

Operator	Description
+ (Addition)	Adds two operands **Ex:** if A=10 B=20 then A + B will give 30
- (Subtraction)	Subtracts the second operand from the first **Ex:** if A=10 B=20 then A - B will give -10
*** (Multiplication)**	Multiply both operands **Ex:** if A=10 B=20 then A * B will give 200
/ (Division)	Divide the numerator by the denominator **Ex:**if A=10 B=20 then B / A will give 2
% (Modulus)	Outputs the remainder of an integer division **Ex:**if A=10 B=20 then B % A will give 0
++ (Increment)	Increases an integer value by one **Ex:**if A=10 then A++ will give 11
--	Decreases an integer value by one

Working with Operators

Comparison Operators

- JavaScript supports the following comparison operators :
  1. Equal(Ex: (A == B))
  2. Not Equal(Ex: (A != B))
  3. Greater than(Ex: (A > B)
  4. Less than(Ex: (A < B))
  5. Greater than or Equal to(Ex: (A >= B))
  6. Less than or Equal to (Ex: (A <= B))

Working with Operators

Logical Operators

- JavaScript supports the following logical operators:
  1. &&(Ex: (x < 10 && y > 1))
  2. ||(Ex: (x == 5 || y == 5))
  3. !(Ex: !(x == y))

Working with Operators

Assignment Operators

- JavaScript supports the following assignment operators

1. = (Simple Assignment)

   Ex: C = A + B will assign the value of A + B into C

2. += (Add and Assignment)

   Ex: C += A is equivalent to C = C + A

3. -= (Subtract and Assignment)

   Ex: C -= A is equivalent to C = C – A

4. *= (Multiply and Assignment)

   Ex: C *= A is equivalent to C = C * A

5. /= (Divide and Assignment)

   Ex: C /= A is equivalent to C = C / A

6. %= (Modules and Assignment)

   Ex: C %= A is equivalent to C = C % A

Working with Operators

Conditional Operator (? :)

- The conditional operator first evaluates an expression for a true or false value and then executes one of the two given statements depending upon the result of the evaluation.
- Syntax:

  ? : (Conditional)

  Conditional ? X : Y

  If Condition is true? Then value X : Otherwise value Y

Creating JavaScript Functions

- A function is **a group of reusable code** which can be called anywhere in your program. This eliminates the need of writing the same code again and again.
- A JavaScript function contains code that will be **executed** by **an event** or by a **call to the function**.
- The **parentheses** may **include parameter** names **separated** by **commas**: (parameter1, paremeter2,..)
- The **code to be executed**, by the function, is placed inside **curly brackets**.

Creating JavaScript Functions

Function Definition

- Define a function in JavaScript is by using the **function** keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.
- Syntax

```
<script type = "text/javascript">
 function functionname(parameter-list) {
        statements
}
</script>
```

Creating JavaScript Functions

- The following example defines a function called sayHello that takes no parameters:

```
<script type = "text/javascript">
 function sayHello() {
Document.write("Hello there");
}
</script>
```

Creating JavaScript Functions

- The following example defines a function called sayHello that takes two parameters:

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

    function sayHello(name, age) {
    document.write (name + " is " + age + " years old.");
    }

 </script>
```

Strings

- A **string** can be defined as a **sequence of letters, digits, punctuation and so on.**
- Strings can be **joined** together with the **+ operator**, which is called **concatenation**.

  For example,

  mystring= "Example of" + "Javascript string";
- As string is an object type it also has some useful features.

  For example,

  lenStr=mystring.**length**;

  which returns the **length** of the **string** in **integer**

Strings

- There are also number of methods available for string.

Method	Description
charAt	Return the character at a specific index
indexOf	Find the first index of a character
lastIndexOf	Find the last index of a character
substring/substr	Return a section of a string
toLowercase	Convert a string to lower case
toUpperCase	Convert a string to upper case

Using Arrays

- An array is a special variable, which can hold more than one value at a time.
  - **var *variable= new Array ();***

    Where variable is the name of the array variable.

    OR

  - **var *array_name* = [*item1*, *item2*, ...];**

Access the Elements of an Array

- You access an array element by referring to the **index number**.
- Example:
  - var students = new Array("Zara", "Sara", "Jay");
  - var name = students[0];
  - This statement accesses the value of the first element in students

Array are objects

- Arrays are a special type of objects. The typeof operator in JavaScript returns "object" for arrays.
- Arrays use **numbers** to access its "elements".
- Objects use **names** to access its "members".
- The Difference Between Arrays and Objects:
  - In JavaScript, **arrays** use **numbered indexes**.
  - In JavaScript, **objects** use **named indexes**.

Array Methods

- Array provides several built-in methods like:
  - length()
  - sort()
  - push()
  - pop()
  - concat()
  - reverse()

JavaScript if else statement

- Conditional statements are used to perform different actions based on different conditions.
- JavaScript have the following conditional statements:
  - Use **if** to specify a block of code to be executed, if a specified condition is true
  - Use **else** to specify a block of code to be executed, if the same condition is false
  - Use **else if** to specify a new condition to test, if the first condition is false

JavaScript if else statement

The if Statement

- Use the if statement to specify a block of JavaScript code to be executed if a condition is true.
- Syntax

```
if (condition) {
    //  block of code to be executed if the condition is true
}
```

JavaScript if else statement

The else Statement

- Use the else statement to specify a block of code to be executed if the condition is false.
- Syntax:

```
if (condition) {
  //  block of code to be executed if the condition is true
} else {
  //  block of code to be executed if the condition is false
}
```

JavaScript if else statement

The else if Statement

- Use the else if statement to specify a new condition if the first condition is false.
- Syntax

```
if (condition1) {
  //  block of code to be executed if condition1 is true
} else if (condition2) {
  //  block of code to be executed if the condition1 is false and condition2 is true
} else {
  //  block of code to be executed if the condition1 is false and condition2 is false
}
```

Working with Program Loops

- A **program 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

- The for loop has the following syntax:

```
for (statement 1; statement 2; statement 3)
{
    // code block to be executed
}
```

- Statement 1 is executed (one time) before the execution of the code block.
- Statement 2 defines the condition for executing the code block.
- Statement 3 is executed (every time) after the code block has been executed.

The For Loop

- Example:

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

For/in loop

- The JavaScript for/in statement loops through the properties of an Object.
- The block of code inside the loop will be executed once for each property.
- The JavaScript for/in statement can also loop over the properties of an Array.
- Syntax for/in over objects:

```
for (variable in object) {
  // code block to be executed
}
```

- Syntax for/in over arrays:

```
for (variable in array) {
  //code
}
```

The While Loop

- The while loop loops through a block of code as long as a specified condition is true.
- Syntax

```
while (condition) {
  // code block to be executed
}
```

The While Loop

- Example
- In the following example, the code in the loop will run, over and over again, as long as a variable (i) is less than 10:

```
while (i < 10) {
  document.write( "The number is " + i);
  i++;
}
```

The Do/While Loop

- The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.
- Syntax

```
do {
    // code block to be executed
}
while (condition);
```

The Do/While Loop

- Example
- The example below uses a do/while loop. The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested:

```
do {
    document.write( "The number is " + i);
     i++;
}while (i < 10);
```

Break and Continue

- The break statement "jumps out" of a loop.
- Example:

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

Break and Continue

- The continue statement "jumps over" one iteration in the loop.
- The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

```
for (i = 0; i < 10; i++) {
      if (i === 3) { continue; }
      document.write("The number is " + i + "<br>“);
    }
```

JavaScript Popup Boxes

- In **Javascript**, **popup boxes** are used to display the message or notification to the user.

JavaScript has three kind of popup boxes:

- Alert box,
- Confirm box, and
- Prompt box.

JavaScript Popup Boxes

- Alert Dialog Box
  - An alert dialog box is mostly used to give a warning message to the users. For example, if one input field requires to enter some text but the user does not provide any input, then as a part of validation, you can use an alert box to give a warning message.
  - However, an alert box can still be used for friendlier messages. Alert box gives only one button "OK" to select and proceed.

JavaScript Popup Boxes

- Confirmation Dialog Box
  - A confirmation dialog box is mostly used to take user's consent on any option. It displays a dialog box with two buttons: **OK** and **Cancel**.
  - If the user clicks on the OK button, the window method **confirm()** will return true. If the user clicks on the Cancel button, then **confirm()** returns false.

JavaScript Popup Boxes

- Prompt Dialog Box
  - The prompt dialog box is very useful when you want to pop-up a text box to get user input. Thus, it enables you to interact with the user. The user needs to fill in the field and then click OK.
  - This dialog box is displayed using a method called **prompt()** which takes two parameters: (i) a label which you want to display in the text box and (ii) a default string to display in the text box.
  - This dialog box has two buttons: **OK** and **Cancel**. If the user clicks the OK button, the window method **prompt()** will return the entered value from the text box. If the user clicks the Cancel button, the window method **prompt()**returns **null**.

JavaScript Events

- JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.

Examples of events

- A mouse click
- A web page or an image loading
- Mousing over a hot spot on the web page
- Selecting an input box in an HTML form
- Submitting an HTML form

Onclick

- The onclick event occurs when the user clicks on an element.
- Syntax
  - <*element* onclick="*myScript*">

Onchange

- The onchange event occurs when the value of an element has been changed.
- For radio buttons and checkboxes, the onchange event occurs when the checked state has been changed.
- Syntax
  - <*element* onchange="*myScript*">

Ondblclick

- The ondblclick event occurs when the user double-clicks on an element.
- Syntax
  - <*element* ondblclick="*myScript*">

Onerror

- The onerror event is triggered if an error occurs while loading an external file (e.g. a document or an image).
- Syntax
  - <*element* onerror="*myScript*">

Onfocus

- The onfocus event occurs when an element gets focus.
- Syntax
  - <*element* onfocus="*myScript*">

Onkeypress

- Execute a JavaScript when a user presses a key.
- Syntax
  - <*element* onkeypress="*myScript*">

JavaScript Events

- onkeyup - A keyboard key is released
- onload - A page or an image is finished loading
- onmousedown - A mouse button is pressed
- onmouseup - A mouse button is released
- onmouseout - The mouse is moved off an element
- onmouseover - The mouse is moved over an element

JavaScript Events

- onreset - The reset button is clicked
- onselect - Text is selected
- onsubmit - The submit button is clicked

JavaScript Object

- In JavaScript, an object is a standalone entity, with properties and method.
- Compare it with a car, for example. A car is an object which has **properties** like weight and color, and **methods** like start and stop.
- All cars have the same **properties**, but the property **values** differ from car to car.
- All cars have the same **methods**, but the methods are performed **at different times**.
- Example: *String JavaScript object has **length** property and **toUpperCase()** method*

JavaScript Object

- JavaScript variables are containers for data values.
- This code assigns a **simple value** (BMW) to a **variable** named car:
  - var car = “BMW";
- Objects are variables too. But objects can contain many values.
- This code assigns **many values** (Fiat, 500, white) to a **variable** named car:
  - var car = {type:"Fiat", model:"500", color:"white"};

JavaScript Object

Properties

- Properties are object attributes.
- Object properties are defined by using the object's name and the property name.

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

– **document** is the object.

– **bgcolor** is the property.

JavaScript Object

Method

- In Javascript method is function that is invoked through object
- **To invoke method**
  - myobject.method();
  - myobject.method(x,y,z);
- Exa.
  - var d= new String();
  - d.toUppercase();

JavaScript's inbuilt Objects

- JavaScript comes with some inbuilt objects which are,
  - String
  - Date
  - Array
  - Boolean
  - Math

  etc....

User Defined Objects

- JavaScript allows you to create your own objects.
- The first step is to use the new operator.

  var person= new Object();
- This creates an empty object.
- This can then be used to start a new object that you can then give new properties and methods.
- Example:
  - person.firstname="Alex"
  - Person.lastname="Shah"
  - Person.age=50;

History Object

- The history object contains the URLs visited by the user (within a browser window).
- Each window maintains a list of recent pages that the browser has visited.
- While the history object's list contains the URLs of recently visited pages, those URLs are not generally accessible by script due to privacy and security limits imposed by browsers.
- Methods of the history object allow for navigating backward and forward through the history relative to the currently loaded page.

History Object

Property	Description
length	Returns the number of URLs in the history list
Method	Description
back()	Loads the previous URL in the history list
forward()	Loads the next URL in the history list
go()	Loads a specific URL from the history list

NAVIGATOR OBJECT

- **NAVIGATOR OBJECT**
- **navigator** : The navigator object contains information about the browser.
- Read-only!

NAVIGATOR OBJECT

Property	Description
appCodeName	Returns the code name of the browser
appName	Returns the name of the browser
appVersion	Returns the version information of the browser
cookieEnabled	Determines whether cookies are enabled in the browser
onLine	Boolean, returns true if the browser is on line, otherwise false.
Platform	Returns for which platform the browser is compiled

mozilla

Nerscape

win32

SCREEN OBJECT

- **screen** : it will give information about size of user's display and color depth.

Property	Description
availHeight	Returns the height of the screen (excluding the Windows Taskbar)
availWidth	Returns the width of the screen (excluding the Windows Taskbar)
height	Returns the total height of the screen
pixelDepth	Returns the color resolution (in bits per pixel) of the screen
width	Returns the total width of the screen

DOCUMENT OBJECT

- The document object represents your web page.
- If you want to access any element in an HTML page, you always start with accessing the document object.
- For example:
  - document.bgcolor=red
  - document.linkcolor=yellow
  - document.write ("<h2> Hello World </h2>);

DOCUMENT OBJECT

- We can access any form information in a document by using the **form name.**
- Example

```
<form name="userdetails">
<input type="text" name="fname"/>
<input type="text" name="lname"/>
<input type="submit" name="submit"/>
</form>
```

- it can be referred by **document.userdetails,** any element within it can be accessed by **document.userdetails.fname.value**

Document Object Properties

Property	Description
anchors	Returns a collection of all the anchors in the document
applets	Returns a collection of all the applets in the document
body	Returns the body element of the document
cookie	Returns all name/value pairs of cookies in the document
domain	Returns the domain name of the server that loaded the document
forms	Returns a collection of all the forms in the document
images	Returns a collection of all the images in the document
links	Returns a collection of all the links in the document
title	Sets or returns the title of the document
URL	Returns the full URL of the document

Document Object Methods

Method	Description
write()	Writes HTML expression or Javascript code to a document
writeln()	Same as write(), but adds a newline character after each statement
open()	Opens an output stream to collect the output from document.write() or document.writeln()
close()	Closes the output stream previously opened with document.open()
getElementById()	Accesses element with a specified id
getElementsByName()	Accesses all elements with a specified name

getElementById()

- When we suppose to get the reference of the element from HTML in Javascript using id specified in the HTML we can use this method

HTML

```
<html>
<body>
<input type="text" id="demo">
</body>
</html>
```

Javascript

```
<script>
function myFunction(){
var txt=document.getElementById("demo")
alert(txt.value);
}
</script>
```

getElementsByName()

- When we suppose to get the reference of the elements from HTML in Javascript using name specified in the HTML we can use this method.
- It will return the array of elements with the provided name.

HTML

```
<html>
<body>
<input type="text" name="myText">
</body>
</html>
```

JavaScript

```
<script>
function myFunction(){
a=document.getElementsByName("myText")[0];
alert(a.value);
}
</script>
```

getElementsByTagName()

- When we suppose to get the reference of the elements from HTML in Javascript using name of the tag specified in the HTML we can use this method.
- It will return the arr
provided tag nam

HTML

```
<html>
<body>
<input type="text" name="Username">
<input type="text" name="Password">
</body>
</html>
```

JavaScript

```
<script>
function myFunction(){
a=document.getElementsByTagName("input")[0];
alert(a.value);
}
</script>
```

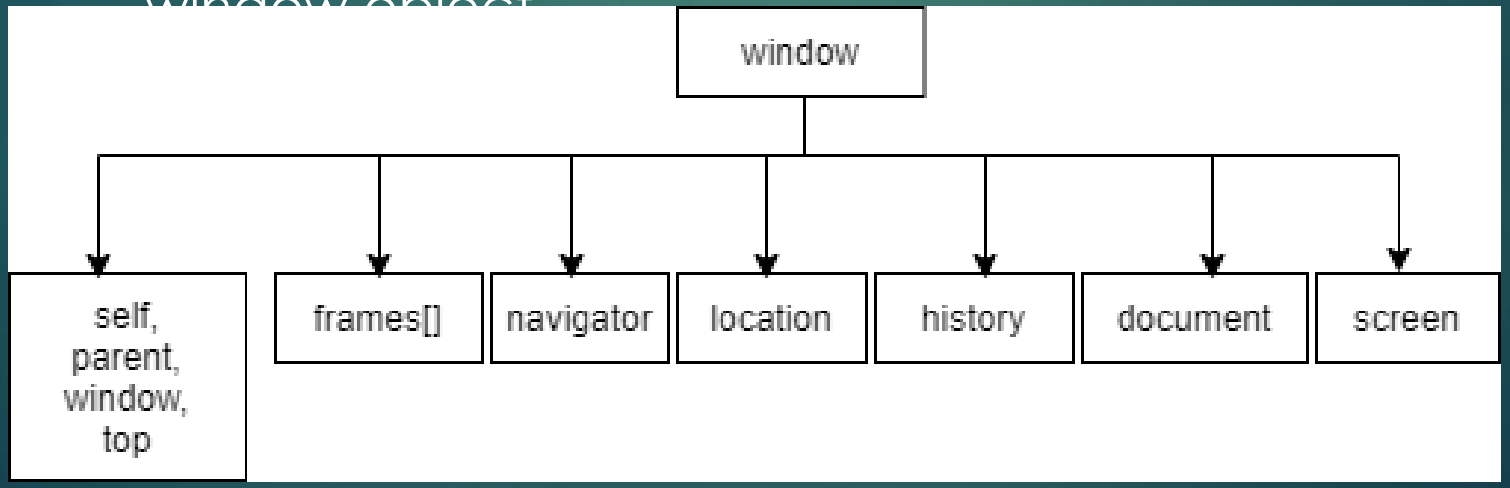
DOM

- Document Object Model (DOM) is a platform and language neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents.
- The window object is the primary point from which most other objects come.
- From the current window object access and control can be given to most aspects of the browser features and the HTML document.
- When we write:
  - document.write("Hello World");
- We are actually writing:
  - window.document.write("Hello World");

  The window is just there by default

DOM

- This window object represents the window or frame that displays the document and is the global object in client side programming for Javascript.
- All the client side objects are connected to the window object



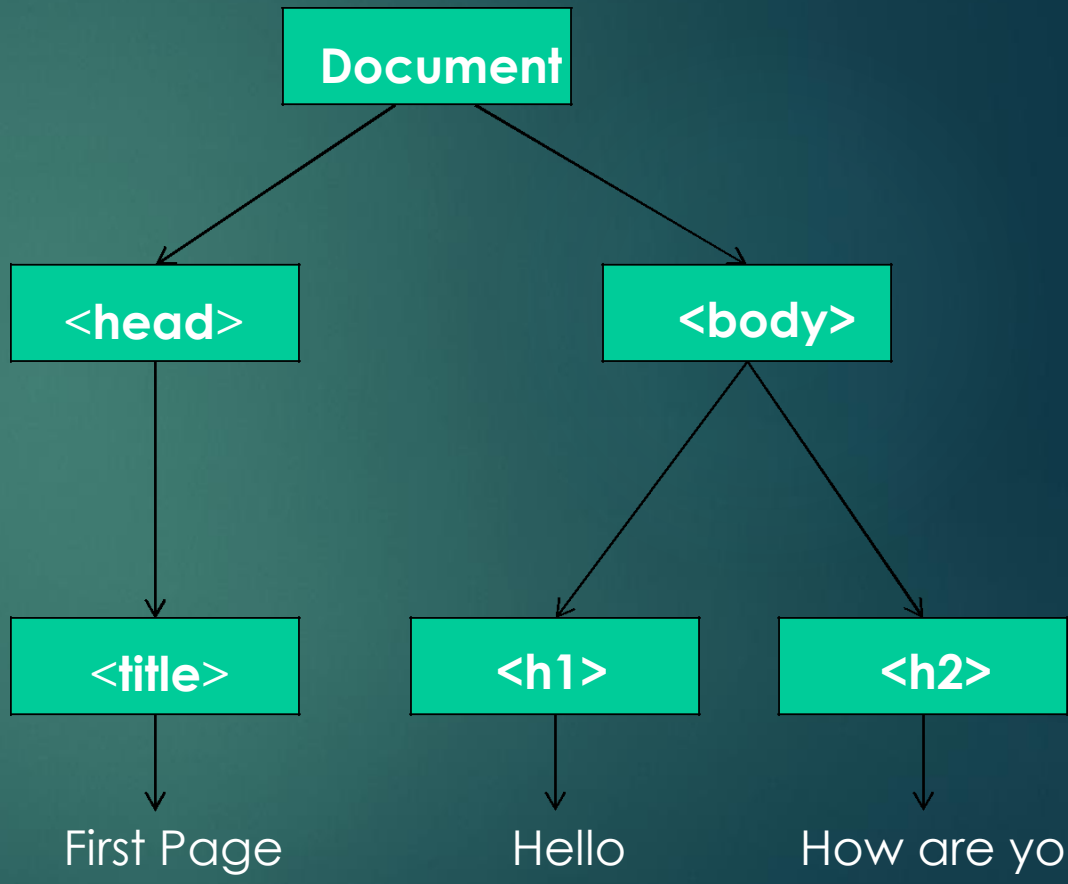
DOM Tree

- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.
- The document in DOM are represented using a tree like structure in which every element is represented as a node.
- **Basic terminologies used in DOM tree as follows:**

1. Every element in the DOM tree is called node.
2. The topmost single node in the DOM tree is called root.
3. Every child node must have parent node.
4. The bottommost nodes that have no children are called leaf nodes.
5. The nodes that have the common parent are called siblings.

DOM Tree

```
<html>
<head>
<title> First Page </title>
</head>
<body>
<h1> Hello </h1>
<h2> How are you </h2>
</body>
</html>
```



Example

```
<html>
<head>
 <title>DOM Tree</title>
</head>
<body>
 <h1>DOM Lesson one</h1>
 <p>Hello world</p>
 <table>
 <tr class="College">
 <td>BVM</td>
 <td>GCET</td>
 <td>ADIT</td>
 </tr>
 </table>
</body>
</html>
```

Forms And Validation

- HTML form validation can be done by JavaScript.
- If a form field is empty or password length is short or password not match with the confirm password, then this function alerts a message, and returns false, to prevent the form from being submitted

Forms And Validation

Required Fields

- It will check if a field has been left empty or not. If the field is blank, an alert box alerts a message, the function returns false, and the form will not be submitted

Forms And Validation

- **Name and Password Validation:**

  It will check name field has not been left empty and password length must be greater than 6 characters

Forms And Validation

- **Validating Password & Confirm password field**

  It will validate password with confirm password

Forms And Validation

- **Validate Number field**

  It will check whether given input is number or not.

Forms And Validation

- **Validation of Radio Button:**

  It will check that the radio button is marked or not.

Validation using RegExp

- A regular expression is an object that describes a pattern of characters.
- To find word character in the string we can use \w
  - We can also use [a-z], [A-Z] or[a-zA-Z] for the same
- To find non-word characters in the string we can use \W
- To find digit characters in the string we can use \d
  - We can also use [0-9] for the same
- To find non-digit characters in the string we can use \D
- We can use \n for newline and \t for tab

RegExp Quantifiers

Quantifier	Description
n+	Matches any string that contains at least one n
n*	Matches any string that contains zero or more occurrences of n
n?	Matches any string that contains zero or one occurrences of n
n$	Matches any string with n at the end of it
^n	Matches any string with n at the beginning of it

Validation using Regular Expression

- **Validation of all letters using Regular expression**

  var letters = /^[A-Za-z]+$/;

  / .. /All regular expressions start and end with forward slashes.

  ^Matches the beginning of the string or line.

- **Validation of letters and numbers using Regular expression**

  var alphanumeric = /^[0-9a-zA-Z]+$/;

- **Validation of password using Regular expression**

  var password= /^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])\w{6,}$/;

Validation using Regular Expression

Validation of email using Regular expression

```
var email= /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/;
```

Explaination:

- **/^[a-zA-Z0-9._-]+**: Means that the email address must begin with alpha-numeric characters (both lowercase and uppercase characters are allowed). It may have periods,underscores and hyphens.
- **@**: There must be a '@' symbol after initial characters.
- **[a-zA-Z0-9.-]+**: After the '@' sign there must be some alpha-numeric characters. It can also contain period ('.') and hyphens('-')
- **\.**: After the second group of characters there must be a period ('.'). This is to separate domain and subdomain names.
- **[a-zA-Z]{2,4}$/**: Finally, the email address must end with two to four alphabets. Having a-z and A-Z means that both lowercase and uppercase letters are allowed. {2,4} indicates the minimum and maximum number of characters. This will allow domain names with 2, 3 and 4 characters e.g.; us, tx, org, com, net, wxyz.

DHTML

- DHTML stands for Dynamic HTML.
- It refers to the web pages that move, animate or respond to the user after download to the browser.
- It describes HTML pages with the dynamic content.
- The DHTML is based on the properties of the HTML, javascript, CSS, and DOM (Document Object Model which is used to access individual elements of a document) which helps in making dynamic content.
- DHTML is not an individual technology. It uses the **JavaScript** technology for accessing, controlling, and manipulating the HTML elements.

Uses of DHTML

- It is used for designing the animated and interactive web pages that are developed in real-time.
- DHTML helps users by animating the text and images in their documents.
- It allows the authors for adding the effects on their pages.
- It is also used to add the ticker on various websites, which needs to refresh their content automatically.

Features of DHTML

- Its simplest and main feature is that we can create the web page dynamically.
- Dynamic Style is a feature, that allows the users to alter the font, size, color, and content of a web page.
- It provides the facility for using the events, methods, and properties. And, also provides the feature of code reusability.
- It also provides the feature in browsers for data binding.
- Using DHTML, users can easily create dynamic fonts for their web sites or web pages.
- With the help of DHTML, users can easily change the tags and their properties.
- The web page functionality is enhanced because the DHTML uses low-bandwidth effect.