PHP

Module 2

Jishna K, CAS Thamarassery

Javascript: Introduction, Client side programming, script tag, comments, variables. Including JavaScript in HTML: head, body, external.

Data types. Operators: Arithmetic, Assignment, Relational, Logical. Conditional Statements, Loops, break and continue. Output functions: write,writeln, popup boxes: prompt, alert, confirm.

Functions: Built-in Global Functions: alert(),prompt(), confirm(), isNan(), Number(), parseInt(). User Defined Functions, Calling Functions with Timer, Events Familiarization: onLoad, onClick, onBlur, onSubmit, onChange, Document Object Model (Concept). Objects: String, Array, Date.

Introduction to Javascript

- JavaScript is the client side object based scripting language used today on the web.

- Javascript helps to create intelligent web pages. ie, pages that verify input,calculate it, and make decisions based on it.

- Javascript is interpreted language, which implies that script written in javascript are processed line by line.

Javascript is used to perform the following tasks :

- Read elements from documents and write elements and text into document.
- validation
- Manipulate or move text
- Create pop up windows
- Perform mathematical calculations on data
- React to events
- Retrieve the current date and time
- Perform actions based on certain conditions

Origins of Javascript

- JavaScript was originally developed by Brendan Eich under the name ***Mocha*** which was later renamed to **Livescript** and then to **JavaScript**.

- Javascript is now officially known as ECMA script.

Including Script in Web pages

- In the head Element
- In the <body>section
- In an external file

A single HTML document can use all 3 types because there is no limit on the number of scripts one document can contain.

https://www.w3schools.com/js/js_whereto.asp

Javascript in the head section

- In HTML, JavaScript code is inserted between `<script>` and `</script>` tags.

- A JavaScript `function` is placed in the `<head>` section of an HTML page.

- The function is invoked (called) when a button is clicked (or an event is occured).

JavaScript in <body>

- A JavaScript function is placed in the <body> section of an HTML page.
- The function is invoked (called) when a button is clicked (or an event is occured).

External JavaScript

- External scripts are practical when the same code is used in many different web pages.

- External 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. Eg:

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

JavaScript Comments (https://www.w3schools.com/js/js_comments.asp)

- Comments can be used to explain JavaScript code, and to make it more readable.

- Comments can also be used to prevent execution, when testing alternative code.

- Types
    - Single Line Comments
    - Multi-line Comments

Single Line Comments

- Single line comments start with //.
- Any text between // and the end of the line will be ignored by JavaScript (will not be executed).
- Eg:

```
// This is a comment.
```

Multi-line Comments

- Multi-line comments start with `/*` and end with `*/`.

- Any text between `/*` and `*/` will be ignored by JavaScript.

  Eg:

```
/* This is a
comment. */
```

JavaScript Variables (https://www.w3schools.com/js/js_variables.asp)

- JavaScript variables are containers for storing data values.
- Var keyword is used for declaring variables.
- To assign a value to the variable, use the equal sign.

Eg:

```
var x = 5;

var y = 6;

var z = x + y;
```

JavaScript Identifiers

- All JavaScript variables must be identified with unique names.

- These unique names are called identifiers.

- Identifiers can be short names (like x,y) or more descriptive names (age, sum).

- Many variables can be declared in one statement.

- Start the statement with `var` and separate the variables by comma.

  Eg:

  ```
  var person = "Arun", course= "bca", rollno = 2;
  ```

- A variable declared without a value will have the value `undefined`.

The general rules for 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(keywords) cannot be used as names.

JavaScript Operators

(https://www.tutorialsteacher.com/javascript/javascript-operators)

(https://www.w3schools.com/js/js_operators.asp)

- Arithmetic operators
- Assignment operators
- Relational operators
- Logical operators
- Conditional operators

Arithmetic operators

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement
**	Exponentiation (ES2016)

Assignment Operators

Operator	Example	same as
● =	x = y	x = y
● +=	x += y	x = x + y
● -=	x -= y	x = x - y
● *=	x *= y	x = x * y
● /=	x /= y	x = x / y
● %=	x %= y	x = x % y
● **=	x **= y	x = x ** y

String Operators

- When used on strings, the + operator is called the concatenation operator.

- The += assignment operator can also be used to add (concatenate) strings.

- Adding two numbers, will return the sum, but adding a number and a string will return a string.

Comparison Operators (Relational)

● >	greater than
● <	less than
● >=	greater than or equal to
● <=	less than or equal to
● ==	equal to
● !=	not equal
● ===	equal value and equal type
● !==	not equal value or not equal type

Logical Operators

&&	logical and
||	logical or
!	logical not

Ternary Operator (conditional operator)

- Ternary operator ?: assigns a value to a variable based on some condition. This is like short form of if-else condition.

- Syntax:

```
<condition> ? <value1> : <value2>;
```

- Eg:

```
var c = (a > b)? a : b;
```

JavaScript Data Types

- JavaScript provides different **data types** to hold different types of values. There are two types of data types in JavaScript.
    - Primitive data type
    - Non-primitive data type

- JavaScript is a dynamic or loosely-typed language because a variable can hold value of any data type at any point of time.

https://www.tutorialsteacher.com/javascript/javascript-data-types
https://www.w3schools.com/js/js_datatypes.asp

Primitive Data Types

- String
- Number
- Boolean
- Null
- Undefined

String

- A string is textual content. It must be enclosed in single or double quotation marks.

Number

- Number type represents integer, float, hexadecimal, octal or exponential value.

Boolean

- Boolean can have only two values, true or false. It is useful in controlling program flow using conditional statements.

null

- null is assigned to a variable to denote that currently that variable does not have any value but it will have later on. A null means absence of a value.
- null is of object type. That is, typeof null will return "object".

undefined

- A variable or an object has an undefined value when no value is assigned before using it. So undefined means lack of value or unknown value.

Non-primitive Data Type

- Object
- Array

Object

- JavaScript objects are written with curly braces `{}`.

- Object properties are written as name:value pairs, separated by commas.

Eg:

```
var person = {firstName:"Athira", lastName:"P", age:20};
```

Arrays

- JavaScript arrays are written with square brackets.

- Array items are separated by commas.

- Array indexes are zero-based, which means the first item is [0], second is [1], and so on.

```
var cars = ["Saab", "Volvo", "BMW"];
```

Conditional Statements
(https://www.w3schools.com/js/js_if_else.asp)

- Conditional statements are used to perform different actions based on different conditions.

- JavaScript supports the following forms of conditional statements.
  - if statement
  - if...else statement
  - if...else if... statement.
  - switch 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
    }
```

The if...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
  }
```

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  }
```

switch Statement

- The `switch` statement is used to perform different actions based on different conditions.

- This is how it works:
  - The switch expression is evaluated once.
  - The value of the expression is compared with the values of each case.
  - If there is a match, the associated block of code is executed.
  - If there is no match, the default code block is executed.

Syntax :

```
switch(expression)
{
     case x:
            // code block
           break;
      case y:
           // code block
            break;
     default:
          // code block
     }
```

The break Keyword

- When JavaScript reaches a `break` keyword, it breaks out of the switch block.
- This will stop the execution of inside the block.

The default Keyword

- The `default` keyword specifies the code to run if there is no case match.

Loops

- Loops can execute a block of code as long as a specified condition is true.

- JavaScript supports different kinds of loops:
    - while
    - do/while
    - for
    - for/in
    - for/of

The while loop

(https://www.w3schools.com/js/js_loop_while.asp)

- 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
    }
```

Eg:

```
var i = 0;
while (i < 10)
     {
          document.write(i);
          i++;
     }
```

The do/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);
```

Eg:

```
var i = 0;
 do
      {
           document.write(i);
           i++;
      }while (i < 10);
```

The for loop

- The for loop loops through a block of code a specified number of times.

- Syntax:

```
for (initialization; condition; updation) {
  // code block to be executed
}
```

Eg:

```
for(i = 0 ; i < 10 ; i++)
    {
        document.write(i);
     }
```

The for/in loop

- The JavaScript for/in statement loops through the properties of an object.

- Syntax:

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

Eg:

```
var person = { fname:"John",  lname:"Doe",  age:25 };
var x;
for (x in person)
{
  document.write(person[x]);
}
```

The for/of loop

- The JavaScript for/of statement loops through the values of an iterable objects such as Arrays, Strings etc.

- Syntax:

```
for (variable of iterable) {
  // code block to be executed
}
```

Eg:

```
var cars = ['BMW', 'Volvo', 'Mini'];

var x;

for (x of cars)

{

  document.write(x + "<br >");

}
```

Javascript Jump statements

- **Jump statements** cause an unconditional **jump** to another statement elsewhere in the code.

- They are used primarily to interrupt switch statements and loops.

- In javascript, there are two jump statements:
    - break statement
    - continue statement

Break statement

- The break statement **"jumps out" of a loop or a switch()** statement.

- The break statement breaks the loop and continues executing the code after the loop (if any)

Eg:

```
for (i = 0; i < 10; i++)
  {
     if (i = = 3)
        break;
     document.write(i);
  }
```

Continue Statement

- 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.

Eg:

```
for (i = 0; i < 10; i++)
  {
    if (i = = 3)
       continue;
    document.write(i);
  }
```

JavaScript Output functions:

document.write()

- Write some text directly to the HTML document.

- Syntax :

```
document.write(exp1, exp2, exp3, ...)
```

Multiple arguments can be listed and they will be appended to the document in order of occurrence

document.writeln()

- This method is similar to write(), only it adds a newline character after each statement.

- Syntax :

```
document.writeln(exp1, exp2, exp3, ...)
```

Multiple arguments can be listed and they will be appended to the document in order of occurrence

JavaScript Popup Boxes

- A popup box is a window that displays a message along with an OK button.

- JavaScript has three kind of popup boxes:
  - Alert box
  - Confirm box
  - Prompt box.

Alert Box

- An alert box is often used to display an alert message while executing the javascript code.

- It is also used to display error messages after validating a form.

- When an alert box pops up, the user will have to click "OK" to proceed.

Syntax

```
window.alert("sometext");
```

- The window.alert() method can be written without the window prefix.

Eg:

```
alert("I am an alert box!");
```

<u>Confirm Box</u>

- A confirm box is often used if you want the user to verify or accept something.

- When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

- If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Syntax

```
window.confirm("sometext");
```

- The window.confirm() method can be written without the window prefix.

Eg:

```
if (confirm("Press a button!"))
    {
        txt = "You pressed OK!";
    }
else
    {
        txt = "You pressed Cancel!";
    }
```

Prompt Box

- A prompt box is often used if you want the user to input a value before entering a page.

- When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.

- If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

<u>Syntax</u>

```
window.prompt("sometext","defaultText");
```

- The window.prompt() method can be written without the window prefix.

- Eg:

```
var person = prompt("Please enter your name");
```

JavaScript Functions

- A Function is a group of statements that perform specific tasks and can be kept and maintained separately from main program.

- Some advantages of using functions are :
    - Reduces repetition of code within a program (code reuse)
    - Makes the code much easier to maintain
    - Makes it easier to eliminate the errors

- A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses ( ).
- Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

- The parentheses may include parameter names separated by commas:

  (parameter1, parameter2, ...)

- The code to be executed, by the function, is placed inside curly brackets: { }

Syntax

```
function name(parameter1, parameter2, parameter3)
  {
     // code to be executed
  }
```

Eg:

```
function display( )
  {     document.write("Hello..!");
   }
```

Returning values from function

- A function can return a value back to the script that called the function as a result using the return statement.

- When JavaScript reaches a return statement, the function will stop executing and control of execution passes back to the calling statement.

- Syntax:

```
return value;
```

Eg:

```
function sum(a,b)
  {
     var c = a + b ;
     return c;
  }
var result= sum(10 , 20);
document.write(result);
```

Variable scope

- A scope refers to an area within which a function and its variables are accessible.
- Scope is divided into two :
  - **Global**

    Specifies that a function or a variable can be called or accessed from anywhere in a program.

  - **Local**

    Specifies that a function or a variable can only be accessed from within the function.

Global Functions

- **alert()**
    - Displays an alert message
- **prompt()**
    - Used to input a value.
- **confirm()**
    - Used to verify or accept something.

- **isFinite()**
    - Check the value passed to it is finite or infinite.
- **isNaN()**
    - Check a value is an illegal number. NaN stands for Not a Number.
- **parseInt()**
    - Parses a string and returns integer value.
- **parseFloat()**
    - Parses a string and returns floating point value.
- **Number()**
    - Converts a value of an object into a number.

Javascript Events

- An event is something that happens when user interact with the web page.

- When an event occurs, a javascript event handler is used to detect them and perform specific task.

- By convention, the name for event handlers always begin with the word “**on**”.

Events can be categorized into four groups:

- Mouse events
- Keyboard events
- Form events
- Document / Window events

Mouse events

- **onclick**
  - The user clicks an HTML element
- **ondblclick**
  - The user double clicks an HTML element
- **onmousedown**
  - user presses a mouse button (but not released) over an element
- **onmouseup**
  - user releases a mouse button over an element

- **onmouseover**
  - The user moves the mouse over an HTML element
- **onmousemove**
  - mouse is moving while it is over an element
- **onmouseout**
  - The user moves the mouse away from an HTML element

Eg:

```
<button onclick="alert('click event');"> Click me </button>

<button onmouseover="alert('mouseover event');"> Over me
</button>

<button onmouseout="alert('mouseout event');"> Mouse out
</button>
```

Keyboard events

- **onkeydown**
    - The event occurs when the user is pressing a key
- **onkeyup**
    - The event occurs when the user releases a key
- **onkeypress**
    - The event occurs when the user presses a key. That is, it will generate a keydown and keyup event.

Eg:

```
<input type="text" onkeydown="alert('keydown event');">

<input type="text" onkeyup="alert('keyup event');">

<input type="text" onkeypress="alert('keypress event');">
```

Form events

- **onfocus**
  - The event occurs when an element gets focus
- **onblur**
  - The event occurs when an element loses focus
- **onsubmit**
  - The event occurs when a form is submitted

- **onreset**
    - The event occurs when a form is reset
- **onselect**
    - The event occurs after the user selects some text

      (for <input> and <textarea>)
- **onchange**
    - The event occurs when the content of a form element, the selection, or the checked state have changed

      (for <input> , <select> , and <textarea>)

Eg:

```
<input type="text" onfocus="this.style.background='red';">

<input type="text" onblur="this.style.background='yellow';">
```

Document / Window events

- **onload**
    - The event occurs when a web page has finished loading in the browser.
- **onunload**
    - The event occurs once a page has unloaded (for <body>)
- **onresize**
    - The event occurs when the document view is resized.

      (minimized or maximized)

Eg:

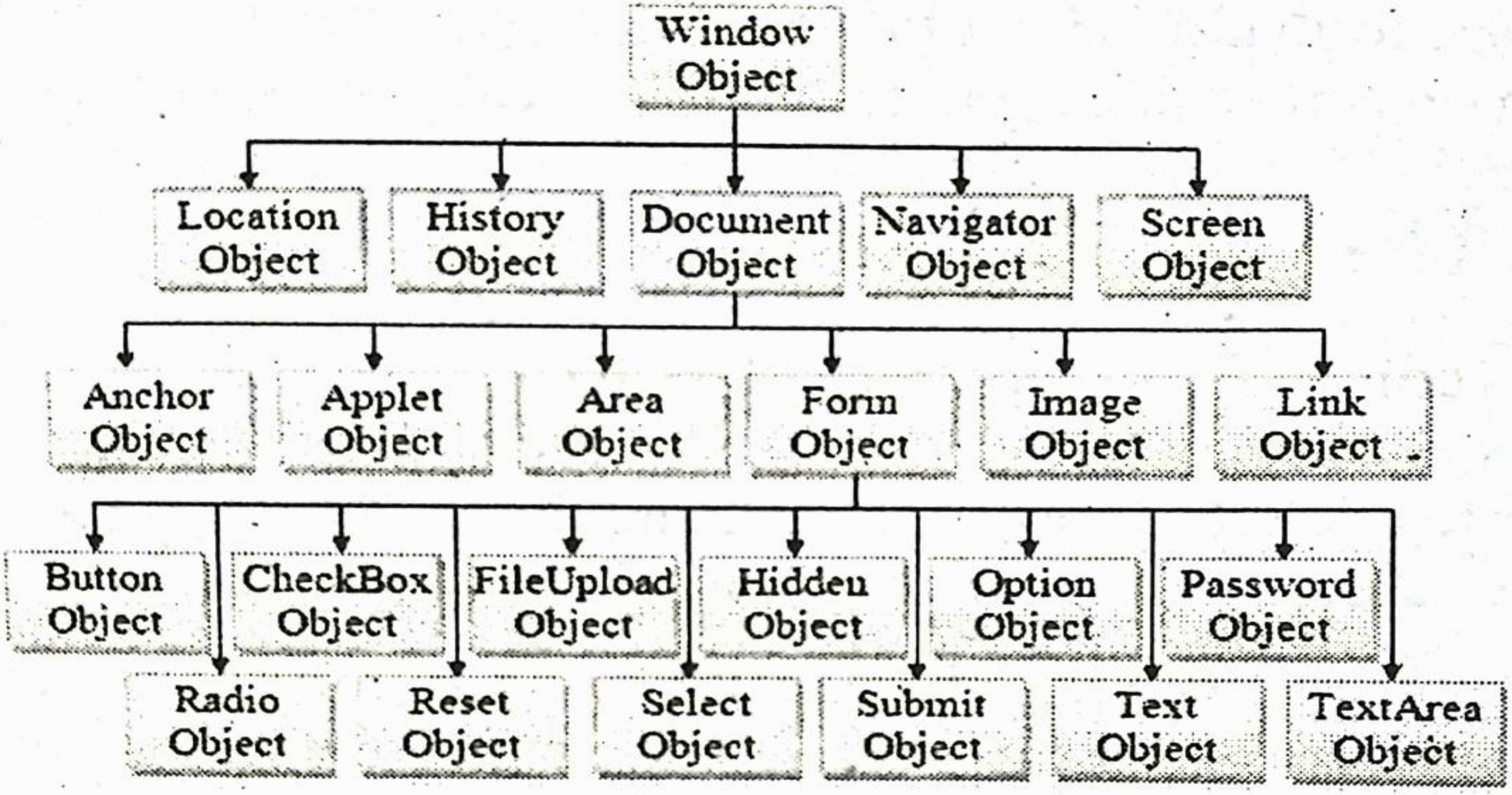
```
<body onload="alert('window is ready');">

<body onunload="alert('window is closing');">

<body onresize="alert('window is resizing');">
```

Document Object Model (DOM)

- DOM is a cross-platform and language independent interface that allows programs and scripts to dynamically access and update the content, structure and style of a documents.

- The DOM is a W3C (World Wide Web Consortium) standard.

- When a web page is loaded, the browser creates a Document Object Model of the page.


Window Object
Location Object
History Object
Document Object
Navigator Object
Screen Object
Anchor Object
Applet Object
Area Object
Form Object
Image Object
Link Object
Button Object
CheckBox Object
FileUpload Object
Hidden Object
Option Object
Password Object
Radio Object
Reset Object
Select Object
Submit Object
Text Object
TextArea Object

DOM Objects

<u>**Window Object**</u> (https://www.w3schools.com/jsref/obj_window.asp)

- At the top of the object hierarchy is the window.

- The window object represents the browser’s frame or window, in which the webpage is contained.

- If a document contain frames, the browser creates one window object for the HTML document, and one additional window object for each frame.

- Via the properties of the window object, user can find out what browser is running, the pages user has visited, the size of the browser window , the size of the user's screen and much more.

- The window object is a global object, which means user don't need to use its name to access its properties and methods.

  Eg:

  ```
  alert("Good Morning");
  ```

History object

(https://www.w3schools.com/jsref/obj_history.asp#:~:text=The%20history%20object%20contains%20the,history%20property.)

- It keeps track of each page that the user visits. This list of pages is commonly called the history stack for the browser.

- It is accessed through the window.history property.

Property :

length	:	Returns the number of URLs in the history list

Method

- 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.

  Eg: history.go(3) will forward 3 pages.

Location object (https://www.w3schools.com/jsref/obj_location.asp)

- The location object contains information about the current URL.

- It contains URL of the page, server hosting the page, port number of the server connection, and the protocol used.

Properties :

● href	-	Sets or returns the entire URL
● hostname	-	Sets or returns the hostname of a URL
● port	-	Sets or returns the port number of a URL
● protocol	-	Sets or returns the protocol of a URL

Navigator object (https://www.w3schools.com/jsref/obj_navigator.asp)

- The navigator object contains information about the browser.

- Using its properties, user can find out name of the browser, version, whether cookies are enabled, language of the browser etc and the operating system the user has.

Screen object (https://www.w3schools.com/jsref/obj_screen.asp)

- The screen object contains information about the visitor's (client’s) screen.

Properties :

- height - Returns the total height of the screen
- width - Returns the total width of the screen
- colorDepth - Returns the bit depth of the color palette for displaying images

Document object (https://www.w3schools.com/jsref/dom_obj_document.asp)

- The document object represents the whole document.

- Via this object user can gain access to the HTML elements, their properties and methods.

- Each of the child objects of the document object represents a collection of similar tags within that document

Important collections are:

● forms	-	contains all the <form> tags in the document.
● images	-	represents all the images in a document.
● links	-	represents all the hyperlinks within a page.
● anchors	-	represents all the anchors in a document.
		(<a> elements with a *name* or *id* attribute rather than *href* attribute.)

Objects, Methods and Properties

Properties of document object

- Cookie
  - It returns all name or value pairs of cookies in the document.

- Domain
  - It returns the domain name of the server that loaded the document.

- lastModified
    - It returns the date and time of last modified document.
- Title
    - It sets or returns the title of the document.
- URL
    - It returns the full URL of the document.

Methods of document object

- open()
  - It opens an output stream to collect the output from document.write()
- close()
  - It closes the output stream previously opened with document.open()
- clear()
  - It clears the document in a window.

- getElementById()
  - It accesses the first element with a specified ID
- getElementByName()
  - It accesses all the elements with a specified name.
- getElementByTagName()
  - It accesses all the elements with a specified tagname.

- write()
  - It writes output to a document.
- writeln()
  - Same as write(), but adds a newline character after each statement.

Built-in Objects

String Objects

- It is used to deal with strings of text.
- To create an instance of string object use new operator.

```
var x = new String(“ good morning ”)
```

Property

- Length - returns the number of characters in a string.

<u>Methods of string object:</u>

- bold()
- italics()
- fontcolor(color)
- fontsize(size)
- CharAt(index)
- big()
- small()
- substring(start,end)
- toLowerCase()
- toUpperCase()

Date Object

- It helps to work with dates and times.

- Date objects are created with the new Date() constructor.

- There are 4 ways to create a new date object:
  - new Date()
  - new Date(year, month, day, hours, minutes, seconds, milliseconds)
  - new Date(milliseconds)
  - new Date(date string)

- **new Date()** creates a new date object with the current date and time:

  Eg:

  ```
  var d = new Date();
  ```

- **new Date(*year, month, ...*)** creates a new date object with a specified date and time.

  Eg:

  ```
  var d = new Date(2018, 11, 24, 10, 33, 30, 0);
  ```

- **new Date(dateString)** creates a new date object from a date string:

  Eg:

  ```
  var d = new Date("October 13, 2014 11:13:00");
  ```

- **new Date(*milliseconds*)** creates a new date object as zero time plus milliseconds:

  Eg:

  ```
  var d = new Date(100000000000);
  ```

Some commonly used methods of Date object:

- getDate()
- getDay()
- getMonth()
- getYear()
- getHours()
- getMinutes()
- getSeconds()
- setDate()
- setHours()
- setMinutes()
- setMonth()
- setSeconds()

Array Object

- Array object is used to handle arrays.

- To create an instance of an array, use new operator along with the array object:

  Eg: `var x=new Array()`

- Array can be filled with values at the time of array creation.

```
var day=new Array("Monday", "Tuesday", "Wednesday")
```

Methods of Array object :

- concat()
  - Joins two or more arrays to create one new one.
- join(separator)
  - Joins all of the elements of an array separated by the character specified as separator.
- reverse()
  - Returns the array reversed.
- slice()
  - Returns a specified part of the array.
- sort()
  - Returns a sorted array.

The End

Thank You

Teacher : Jishna K

College of Applied Science, Thamarassery.