OVERVIEW OF JAVASCRIPT

Introduction - Core Features - Data Types and Variables - Operators, Expressions, and Statements Functions - Objects - Array, Date and Math Related Objects - Document Object Model - Event Handling - Controlling Windows & Frames and Documents – Form Validations

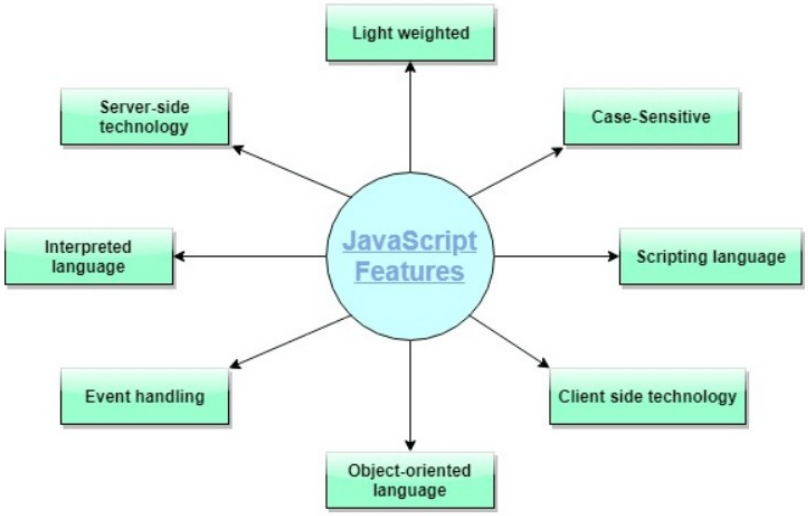
---

Introduction

- JavaScript is the programming language of the Web.
- It is a lightweight, interpreted programming language.
- JavaScript is very easy to implement because it is integrated with HTML
- JavaScript can execute not only in the browser, but also on the server, or on any device that has the JavaScript engine.
- The browser has an embedded engine sometimes called a "JavaScript virtual machine".
- The engine in the browser reads ("parses") the script.
- Then it converts ("compiles") the script to the machine language.
- And then the machine code runs, pretty fast.

Core Features

- The features of Javascript
  - Validating User's Input for errors and also saves time.
  - Simple Client-side Calculations can be performed on the browser.
  - JavaScript provides greater control to the browser rather than being completely dependent on the web servers.
  - It can dynamically generate HTML content for the web, change the existing content, modify styles.
  - React to user actions, run on mouse clicks, pointer movements, key presses.
  - Send requests over the network to remote servers, download and upload files.
  - Get and set cookies, ask questions to the visitor, show messages.
  - Remember the data on the client-side ("local storage").



- JavaScript can be implemented using JavaScript statements that are placed within the <script>... </script> HTML tags in a web page.
- Normally it is used within the <head> tags, If we want to have a script run on some event
- If we need a script to run as the page loads, then the script goes in the <body> portion of the document.
- The script tag takes two attributes
  - **Language**
    - Specifies scripting language that is used.
    - Normally the value will be javascript.
  - **Type**
    - To indicate the scripting language in use and its value should be set to "text/javascript".
- So your JavaScript segment will look like

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

     JavaScript code

     </script>

</head>
```

Example

```
<html>
<head>
<script type = "text/javascript">
function sayHello() {
alert("Hello World")
}
</script>
</head>

<body>
<input type = "button" onclick = "sayHello()" value = "Say Hello" />
</body>
</html>
```

Data Types and Variables

JavaScript allows you to work with three primitive data types

- **Numbers**, eg. 123, 120.50 etc.
- **Strings of text** e.g. "This text string" etc.
- **Boolean** e.g. true or false.
- JavaScript also defines two trivial data types, **null** and **undefined**, each of which defines only a single value.
- It also supports a composite data type known as **object**.
- A variable is a “named storage” for data.
- Variables are declared with the **var** keyword.
- JavaScript also uses the keywords **let** and **const** to declare variables.

Numbers

- JavaScript does not make a distinction between integer values and floating-point values.
- All numbers in JavaScript are represented as floating-point values.
- When a number appears directly in a JavaScript program, it’s called a numeric literal.
- In a JavaScript, a base-10 integer is written as a sequence of digits. For **example**:
  0
  3
  10000000
- JavaScript recognizes hexadecimal (base-16) values.
- A hexadecimal literal begins with “0x” or “0X”, followed by a string of hexadecimal digits.
  example:
  0xff // 15*16 + 15 = 255 (base 10)
  0xCAFE911

- some implementations of JavaScript allow you to specify integer literals in octal (base-8) format.
- An octal literal begins with the digit 0 and is followed by a sequence of digits, each between 0 and 7.
  example:
  0377
  // 3*64 + 7*8 + 7 = 255 (base 10)

Floating-Point Literals

- Floating-point literals can have a decimal point; they use the traditional syntax for real numbers.
- A real value is represented as the integral part of the number, followed by a decimal point and the fractional part of the number.
  example:
  3.14
  2345.789

```
.333333333333333333
6.02e23             // 6.02 × 10 23
1.4738223E-32       // 1.4738223 × 10 −32
```

String Literals

- To include a string literally in a JavaScript program, simply enclose the characters of the string within a matched pair of single or double quotes (' or ").
- Double-quote characters may be contained within strings delimited by single-quote characters, and
- single-quote characters may be contained within strings delimited by double quotes.

examples:

```
"" // The empty string: it has zero characters
'testing'
"3.14"
'name="myform"'
"Wouldn't you prefer O'Reilly's book?"
"This string\nhas two lines"
"π is the ratio of a circle's circumference to its diameter"
```

Working with Strings

- In JavaScript strings can be concatenated using + operator.
- When used on strings, it joins both the strings.

example:

```
msg = "Hello, " + "world";
greeting = "Welcome to my blog," + " " + name;
```

Boolean Values

- A boolean value represents truth or falsehood, on or off, yes or no.
- There are only two possible values of this type.
- The reserved words true and false evaluate to these two values.
- Boolean values are generally the result of comparisons you make in your JavaScript programs.

example:

```
if (a == 7)
    b = b + 1;
else
    a = a + 1;

if (O !== null) ...
```

null and undefined

- **null** is a language keyword that evaluates to a special value that is usually used to indicate the absence of a value.
- Using the typeof operator on null returns the string “object”, indicating that null can be thought of as a special object value that indicates “no object”.

- In practice, however, null is typically regarded as the sole member of its own type, and it can be used to indicate “no value” for numbers and strings as well as objects.
- The undefined value represents a deeper kind of absence.
- It is the value of variables that have not been initialized

Example

```
<scripttype="text/javascript">
var money;
var name;


</script>
```

we can also declare multiple variables with the same var keyword

```
<scripttype="text/javascript">

var money, name;
</script>
```

- Storing a value in a variable is called variable initialization.
- Variable initialization can be done at the time of variable creation or at a later point.

```
<scripttype="text/javascript">
var name ="Ajith";
var cash;
cash=2000.50;


</script>
```

The Global Object

- The properties of this object are the globally defined symbols that are available to a JavaScript program.
- When the JavaScript interpreter starts, it creates a new global object and gives it an initial set of properties that define:

Operators, Expressions, and Statements Functions

Operators

JavaScript supports the following types of operators

- Arithmetic Operators
- Comparison Operators
- Logical Operators
- Assignment Operators
- Bitwise Operators
- Conditional (or ternary) Operators

Arithmetic Operators

Arithmetic operators are used to perform mathematical operations between numeric operands.

Operator	Description	Example
+	Addition	10+20 = 30
-	Subtraction	20-10 = 10
*	Multiplication	10*20 = 200
/	Division	20/10 = 2
%	Modulus (Remainder)	20%10 = 0
++	Increment	var a=10; a++; Now a = 11
--	Decrement	var a=10; a--; Now a = 9

Comparison Operators

comparison operators that compare two operands and return a boolean

value **true** or **false**.

Operator	Description	Example
==	Is equal to	10==20 = false
===	Identical (equal and of same type)	10==20 = false
!=	Not equal to	10!=20 = true
!==	Not Identical	20!==20 = false
>	Greater than	20>10 = true
>=	Greater than or equal to	20>=10 = true
<	Less than	20<10 = false
<=	Less than or equal to	20<=10 = false

Logical operators

Logical operators are used to combine two or more conditions. JavaScript provides the following logical operators.

Operator	Description	Example
&&	Logical AND	(10==20 && 20==33) = false
||	Logical OR	(10==20 || 20==33) = false

!	Logical Not	!(10==20) = true

Assignment Operators

JavaScript provides the assignment operators to assign values to variables with less key strokes. The following operators are known as JavaScript assignment operators.

Operator	Description	Example
=	Assign	10+10 = 20
+=	Add and assign	var a=10; a+=20; result a = 30
-=	Subtract and assign	var a=20; a-=10; result a = 10
*=	Multiply and assign	var a=10; a*=20; result a = 200
/=	Divide and assign	var a=10; a/=2; result a = 5
%=	Modulus and assign	var a=10; a%=2; result a = 0

Bitwise Operators

The bitwise operators perform bitwise operations on operands. The bitwise operators are as follows

Operator	Description	Example
&	Bitwise AND	5 & 1 result 1
|	Bitwise OR	5 | 1 result 1
^	Bitwise XOR	5 ^ 1 result 4
~	Bitwise NOT	~ 5 result 10

<<	Bitwise Left Shift	2<<1 result 4
>>	Bitwise Right Shift	2>>1 result 1
>>>	Bitwise Right Shift with Zero	2>>>1 result 1

Ternary Operator

JavaScript provides a special operator called ternary operator **:?** that assigns a value to a variable based on some condition.

Syntax:

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

- The ternary operator starts with conditional expression followed by the ? operator.
- The second part (after ? and before :) will be executed if the condition turns out to be true.
- If, the condition returns false, then the third part (after :) will be executed.

Example:

```
var a = 10, b = 5;
var c = a > b? a : b;
```

Expressions

Expressions are units of code that can be evaluated and resolve to a value. Expressions in JS can be divided in categories.

Arithmetic Expressions - evaluates to a numeric value.
Example: 10+13;

String Expressions - expressions that evaluate to a string
Example: 'hello' + 'world';

Logical Expressions - Evaluate to the boolean value true or false.
Example: a===20 && b===30;

Primary Expressions - stand alone expressions involving variable, literals and constants.
Example:

```
'hello world';     // A string literal
23;                      // A numeric literal
true;              // Boolean value true
```

Left-hand-side Expressions - Exressions that can appear on the left side of an assignment expression.
Example: i = 10;

Assignment Expressions - Expressions use the = operator to assign a value to a variable.
Example: average = 55;

Statements

- A statement is an instruction to perform a specific action.
- Such actions include creating a variable or a function, looping through an array of elements, evaluating code based on a specific condition etc.
- JavaScript programs are actually a sequence of statements.
- Statements in JavaScript can be classified into the following categories.

Declaration Statements

- Statements which create variables and functions by using the **var** and **function** statements respectively.
- The **var** statement declares a variable or variables.

Example

```
var total = 0;
```

- The **function** keyword is used to define functions.
- Every function must have a funtion name that names the function being declared.
- The function name is followed by a comma-separated list of parameter names in parentheses.
- These identifiers can be used within the body of the function to refer to the argument values passed when the function is invoked.

Example

```
function greet(message) {
        console.log(message);
}
```

Expression Statements

- The simplest kinds of statements in JavaScript are expressions that have side effects.
- Assignment statements are one major category of expression statements.

Example

```
b = 4+38;
```

Compound and Empty Statements

- A statement block combines multiple statements into a single compound statement.
- A statement block is simply a sequence of statements enclosed within curly braces.

Example

```
{
        x = Math.PI;
        cx = Math.cos(x);
        console.log("cos(π) = " + cx);
}
```

- The empty statement allows you to include no statements.
- The empty statement looks like this

Example

A loop which does nothing with an empty statement

```
for(i = 0; i < a.length; a[i++] = 0) ;
```

Conditional Statements

- Conditional statements execute statements based on the value of an expression.
- The **if statement** is the fundamental control statement that allows JavaScript to make decisions, or, more precisely, to execute statements conditionally.

Example

```
if (n == 1)
        console.log("You have 1 new message.");
else
        console.log("You have " + n + " new messages.");
```

- **The switch statement** is used to to perform a multiway branch.

Example

```
switch(n) {
        case 1:
                // Execute code block #1.
                break;

        case 2:
                // Execute code block #2.
                break;

        case 3:
                // Execute code block #3.
                break;

        default:
                // Execute code block #4.
                break;

}
```

Loop statements

- The looping statements are those that bend that path back upon itself to repeat portions of your code.
- Some the looping statments are listed below
- The **while statement**

Example

```
var count = 0;
while (count < 10)
{
    console.log(count);
    count++;
}
```

- The **do/while statement**

Example

```
do {
        console.log(a[i]);
} while (++i < len);
```

- The **for statement** provides a looping construct that is often more convenient

Example

```
for(var count = 0; count < 10; count++)
        console.log(count);
```

JavaScript Objects

- JavaScript object is a non-primitive data-type that allows you to store multiple collections of data.
- A javaScript object is an entity having properties and method
- JavaScript is an object-based language.
- Everything is an object in JavaScript.
- There are 3 ways to create objects.
  1. By object literal
  2. By creating instance of Object directly
  3. By using an object constructor

Object by object literal

- The easiest way to create an object is to include an object literal.
- Each member of an object is a key: value pair separated by commas and enclosed in curly braces {}

Example

```
<script>
        var point = { x:0, y:0 };
        emp={id:102,name:"Senthil Kumar",salary:40000};
</script>
```

By creating instance of Object directly

- Object in javascript can be creted object directly is given below

Example

```
<script>
        var emp=new Object();

        emp.id=101;
        emp.name="Mohammed";
        emp.salary=50000;
        document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
```

By using an Object constructor

- The new operator creates and initializes a new object.
- The new keyword must be followed by a function invocation.
- A function used in this way is called a constructor and serves to initialize a newly created object.

Example

```
<script>
function emp(id,name,salary){
        this.id=id;
        this.name=name;
        this.salary=salary;
}
e=new emp(103,"Raja Ram",30000);

document.write(e.id+" "+e.name+" "+e.salary);
</script>
```

Accessing Object Properties

- Properties of the objects can be accessed using the dot notation.

Example

```
const person = {
   name: 'Agastus',
   age: 20,
};

// accessing property
console.log(person.name);
```

- Another way to access properties is by using bracket Notation, here is the syntax of the bracket notation.

Example

```
const person = {
    name: 'John',
    age: 20,
};

// accessing property
console.log(person["name"]);
```

JavaScript Object Methods

There are various methods of Objects some of them are listed below

S.No	Methods	Description
1	Object.assign()	This method is used to copy enumerable and own properties from a source object to a target object
2	Object.create()	This method is used to create a new object with the specified prototype object and properties.
3	Object.freeze()	This method prevents existing properties from being removed.
4	Object.values()	This method returns an array of values.
5	Object.seal()	This method prevents new properties from being added and marks all existing properties as non-configurable.

Array, Date and Math Related Objects

Date

- The JavaScript date object can be used to get year, month and day.
- The Date object is used to work with dates and times.
- The Javascript Date object can be used to display a timer on the webpage.
- We can create a date using the Date object by calling the new Date() constructor.
- There are four ways of instantiating a date with new Date() constructor
    1. Date()
    2. Date(milliseconds)
    3. Date(dateString)
    4. Date(year, month, day, hours, minutes, seconds, milliseconds)

Example

```
<!DOCTYPE html>
<html>
<body>

<h1>JavaScript </h1>
<h1>Displaying date using Date( ) constructor</h2>
<p id="demo"></p>
<script>
    const d = new Date("11-Dec-2021");
    document.getElementById("demo").innerHTML = d;
 </script>
</body>
</html>
```

JavaScript Date Methods

Some of the JavaScript date methods which are used are listed below.

Method	Description
getDate()	Returns the day of the month (from 1-31)
getDay()	Returns the day of the week (from 0-6)
getFullYear()	Returns the year
getMonth()	Returns the month (from 0-11)
setDate()	Sets the day of the month of a date object
toString()	It returns the date in the form of string.

Math

- The JavaScript math object provides several constants and methods to perform mathematical operation.
- Unlike other global objects, Math does not have a constructor.
- All the properties and methods of Math are static and can be called by using Math as an object without creating it.

Math Properties (Constants)

- The syntax for any Math property is : Math.property.
- JavaScript provides 8 mathematical constants that can be accessed as Math properties

S. No.	Property	Description
1.	E	Returns Euler's number (approx. 2.718)
2.	PI	Returns PI (approx. 3.14)
3.	SQRT2	Returns the square root of 2 (approx. 1.414)
4.	SQRT1_2	Returns the square root of 1/2 (approx. 0.707)
5.	LN2	Returns the natural logarithm of 2 (approx. 0.693)
6.	LN10	Returns the natural logarithm of 10 (approx. 2.302)
7.	LOG2E	Returns the base-2 logarithm of E (approx. 1.442)
8.	LOG10E	Returns the base-10 logarithm of E (approx. 0.434)

Number to Integer

There are 4 common methods to round a number to an integer:

- Math.round(n) Returns n rounded to its nearest integer
- Math.ceil(n) Returns n rounded up to its nearest integer
- Math.floor(n) Returns n rounded down to its nearest integer
- Math.trunc(n) Returns the integer part of n

Math.round(n)

- The JavaScript math.round(n) method returns the rounded integer nearest for the given number.
- If fractional part is equal or greater than 0.5, it goes to upper value 1 otherwise lower value 0.

Example:

```
Math.round(4.6); the result will be 5
Math.round(4.5); the result will be 5
Math.round(4.4); the result will be 4
```

Math.ceil(n)

- The JavaScript math.ceil(n) method returns the largest integer for the

given number, whic will be the rounded up value. For example 4 for 3.7, 6 for 5.9 etc.

Example:

```
Math.ceil(4.9); the result will be 5
Math.ceil(4.2); the result will be 5
```

Math.floor(n)

- The JavaScript math.floor(n) method returns the lowest integer for the given number. For example 3 for 3.7, 5 for 5.9 etc.

  Example:

```
Math.ceil(4.9); the result will be 4
Math.ceil(4.2); the result will be 4
```

Math.random()

- The JavaScript math.random() method returns the random number between 0 to 1.
- **Example**: Math.random()

JavaScript Math Methods

There are various other methods which are used

Method	Description
abs(x)	Returns the absolute value of x
cos(x)	Returns the cosine of x (x is in radians)
log(x)	Returns the natural logarithm (base E) of x
max(x, y, z, ..., n)	Returns the number with the highest value
min(x, y, z, ..., n)	Returns the number with the lowest value
pow(x, y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
sign(x)	Returns if x is negative, null or positive (-1, 0, 1)
sin(x)	Returns the sine of x (x is in radians)
sqrt(x)	Returns the square root of x