INTERNET TECHNOLOGIES

UNIT 1

INTRODUCTION TO JAVASCRIPT

BASICS OF JAVASCRIPT

Javascript

- JavaScript is used to create client-side dynamic pages.
- JavaScript is *an object-based scripting language* which is lightweight and cross-platform.
- JavaScript is not a compiled language, but it is a translated language. The JavaScript Translator (embedded in the browser) is responsible for translating the JavaScript code for the web browser.
- JavaScript is a lightweight, interpreted programming language. It is designed for creating network-centric applications. It is complimentary to and integrated with Java. JavaScript is very easy to implement because it is integrated with HTML.

Why to Learn Javascript

- Javascript is the most popular programming language in the world and that makes it a programmer's great choice. Once you learnt Javascript, it helps you developing great front-end as well as back-end softwares.
- Javascript is everywhere, it comes installed on every modern web browser and so to learn Javascript we really do not need any special environment setup. For example Chrome, Mozilla Firefox , Safari and every browser we know as of today, supports Javascript.
- Javascript helps we create really beautiful and fast websites. We can develop our website with the best Graphical User Experience.
- JavaScript usage has now extended to mobile app development, desktop app development, and game development. This opens many opportunities for the Javascript Programmer.

A simple Javascript code

To Hello World using Javascript

```
<html>
  <body>
    <script language = "javascript" type = "text/javascript">
            document.write("Hello World!")
    </script>
  </body>
</html>
```

Save it as abc.html anywhere on your system. Double click on it. A web page will open up displaying “Hello World”

What is Javascript?

- JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.
- JavaScript was first known as **LiveScript,** but Netscape changed its name to JavaScript, possibly because of the popularity of Java.

Javascript is :

- Lightweight, interpreted programming language.
- Designed for creating network-centric applications.
- Complementary to and integrated with Java.
- Complementary to and integrated with HTML.
- Open platform

Javascript can be divided into 3 parts

1. **Core**
2. **Client side**
3. **Server side**

1. **Core :** It is the heart of the language that includes operators, statements, expressions and sub programs.
2. **Client side :** It is the collection of objects that support control of the browser and interaction with the browser.
3. **Server side :** It is a collection of objects that make a language useful on the web browser.

Features of Javascript

There are following features of JavaScript:

- All popular web browsers support JavaScript.
- JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.
- JavaScript is a weakly typed language, where certain types are implicitly cast (depending on the operation).
- JavaScript is an object-oriented programming language that uses prototypes rather than using classes for inheritance.
- It is a light-weighted and interpreted language.
- It is a case-sensitive language.
- JavaScript is supportable in several operating systems including Windows, macOS, etc.
- It provides good control to the users over the web browsers.

Advantages of Javascript

The merits of using JavaScript are −

- **Less server interaction** − You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
- **Immediate feedback to the visitors** − They don't have to wait for a page reload to see if they have forgotten to enter something.
- **Increased interactivity** − You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
- **Richer interfaces** − You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

Some other advantages include:

- Client-side validation,
- Dynamic drop-down menus,
- Displaying date and time,
- Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box),
- Displaying clocks etc.

Limitations of JavaScript

We cannot treat JavaScript as a full-fledged programming language. It lacks the following important features −

- Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
- JavaScript cannot be used for networking applications because there is no such support available.
- JavaScript doesn't have any multi-threading or multiprocessor capabilities.

JavaScript - Placement in HTML File

We can place the javascript file in the following sections :

- **In the <head>...</head> section.**
- **In the <body>...</body> section.**
- **In an external file and then include in <head>...</head> section.**

1. JavaScript in <head>...</head> section

```
<html>
  <head>
    <script type = "text/javascript">
       document.write(“Hello”);
    </script>
  </head>
    <body>
      <center> Welocme to Javascript Programming</center>
    </body>
</html>
```

2. JavaScript in <body>...</body> section

```
<html>
  <head>
  </head>
  <body>
     <center> Welocme to Javascript Programming</center>
      <script type = "text/javascript">
       document.write(“Hello”);
      </script>
  </body>
</html>
```

3. In an external file and then include in <head>...</head> section.

1. document.write(“hello”);

Type code 1 in notepad and save the notepad with the extension .js ie. for ex- abc.js. Then type the following code in another notepad

2.
```
<html>
  <head>
   <script type="text/javascript" src=“abc.js"></script>
  </head>
  <body>
    <p>Welcome to JavaScript</p>
  </body>
</html>
```

Now type code 2 in notepad and save the notepad with the extension .html ie. for ex- abc.html. Then double click on abc.html. The external script will be embedded during runtime and it will give the desired output

General Syntactic Characteristics of Javascript

- Javascript codes are embedded either directly or indirectly inside html document.
- The type attribute of script must be set “text/javascript”.
- Javascript script can be directly embedded inside a HTML document using the src attribute.

  <script type = “text/javascript” src =“ abc.js”>
- In javascript, identifiers or names are similar to that of programming languages.
- Javascript has a number of reserved keywords. Some of these are break, var, catch, continue, for, if, while, else etc.
- **Whitespace and Line Breaks:** JavaScript ignores spaces, tabs, and newlines that appear in JavaScript programs. We can use spaces, tabs, and newlines freely in our program.

General Syntactic Characteristics of Javascript contd…

- **Semicolons are Optional :** Simple statements in JavaScript are generally followed by a semicolon character, just as they are in C, C++, and Java. JavaScript, however, allows you to omit this semicolon if each of your statements are placed on a separate line. For example, the following code could be written without semicolons.

```
<script language = "javascript" type = "text/javascript“>
  var1 = 10
  var2 = 20
</script>
```

But when formatted in a single line as follows, you must use semicolons −

```
<script language = "javascript" type = "text/javascript“>
  var1 = 10; var2 = 20
</script
```

- **Case Sensitivity :**JavaScript is a case-sensitive language. So the identifiers **Time** and **TIME** will convey different meanings in JavaScript.
- **Comments in JavaScript:**
  a. Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript.
  b. Any text between the characters /* and */ is treated as a comment. This may span multiple lines.

Javascript Variables

A **JavaScript variable** is simply a name of storage location. There are two types of variables in JavaScript : local variable and global variable.

There are some rules while declaring a JavaScript variable (also known as identifiers).

- Name must start with a letter (a to z or A to Z), underscore(_), or dollar($) sign.
- After first letter we can use digits (0 to 9), for example value1.
- JavaScript variables are case sensitive, for example x and X are different variables.

Correct JavaScript variables

- var x = 10;
- var _value="amit";

Incorrect JavaScript variables

- var 123=30;
- var *aa=320;

Example of Javascript

Let’s see a simple example of JavaScript variable.

```
<script>
 var x = 10;
 var y = 20;
 var z=x+y;
 document.write(z);
</script>
```

Javascript Variable Scope

The scope of a variable is the region of your program in which it is defined. JavaScript variables have only two scopes.

- **Global Variables** − A global variable has global scope which means it can be defined anywhere in your JavaScript code.

- **Local Variables** − A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

Javascript Local Variable

A JavaScript local variable is declared inside block or function. It is accessible within the function or block only. For example:

```
<script>
  function abc()
  {
   var x=10;//local variable
  }
</script>
```

Or,

```
<script>
  If(10<13)
 {
 var y=20;//JavaScript local variable
 }
</script>
```

Javascript Global Variable

A **JavaScript global variable** is accessible from any function. A variable i.e. declared outside the function or declared with window object is known as global variable. For example:

```
<script>
 var data=200; //global variable
 function a()
 {
 document.write(data);
 }
 function b()
 {
 document.write(data);
 }
 a(); //calling JavaScript function
 b();
</script>
```

TO FIND FACTORIAL

```
<HTML>
  <HEAD>
    <TITLE> Factorial of a number</TITLE>
   </HEAD>
   <BODY>
   <SCRIPT type = “text/javascript”>
    var a=5,fact=1;
    for(i=1;i<=5,i++)
    {
     fact=fact*i;
    }
    document.write(“factorial is”+fact);
</SCRIPT>
</BODY>
</HTML>
```

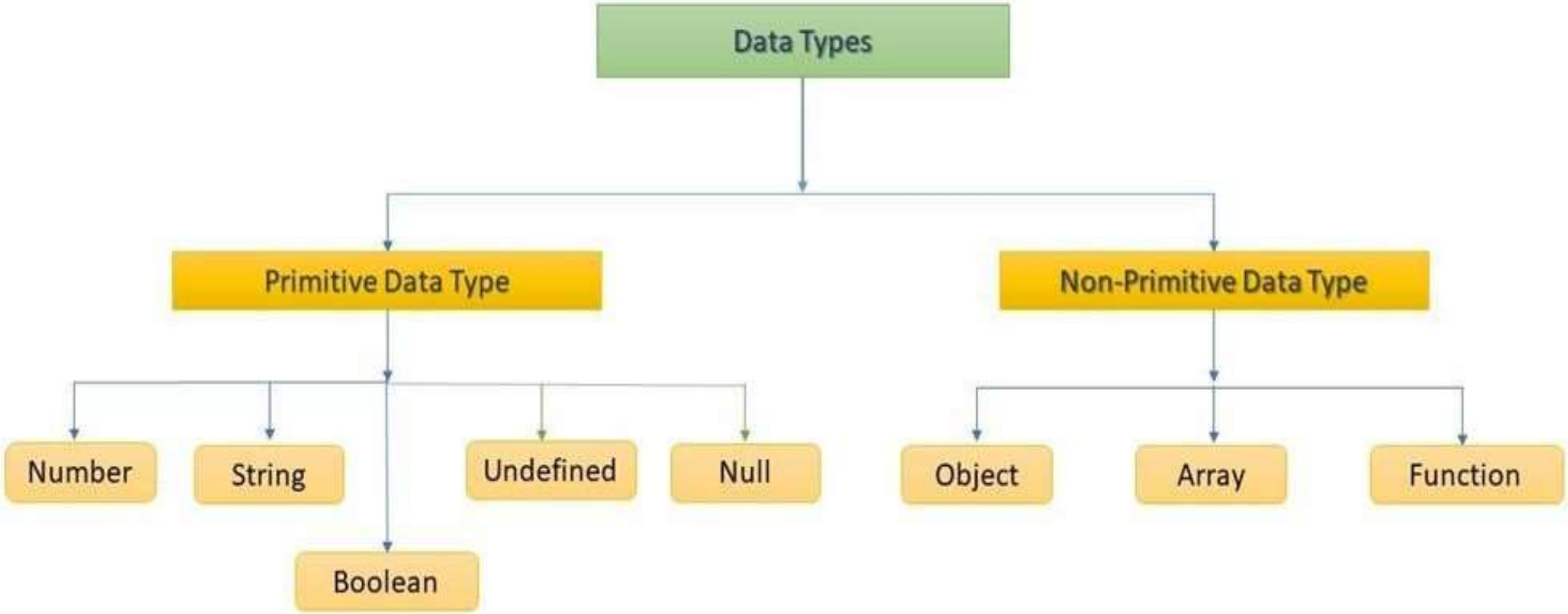
Javascript Reserved Keywords

Keywords are tokens that have special meaning in JavaScript

A list of all the reserved words in are given in the following cannot be used as JavaScript functions, methods, loop labels, or any object names.

abstract	else	instanceof	Switch
boolean	enum	int	Synchronized
break	export	interface	This
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	True
class	float	package	Try
const	for	private	typeof
continue	function	protected	Var
debugger	goto	public	Void
default	if	return	volatile
delete	implements	short	while
do	import	static	With
double	in	super	

Javascript Data Types



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 (reference) data type**

JavaScript is a **dynamic type language**, means you don't need to specify type of the variable because it is dynamically used by JavaScript engine. You need to use **var** here to specify the data type. It can hold any type of values such as numbers, strings etc. For example:

- var a=40;//holding number
- var b="Rahul";//holding string

Primitive Data Type

A **primitive** is a fundamental data type that cannot be broken down into a more simple **data type**. There are five types of primitive data types in JavaScript. They are as follows:

Data Type	Description
String	represents sequence of characters e.g. “hello” or ‘hello’
Number	represents numeric values e.g. 100, 8.5
Boolean	represents boolean value either false or true
Undefined	represents undefined value
Null	represents null i.e. no value at all

Simple Program on Primitive Data Type

```
<HTML>
  <HEAD>
    <TITLE> Factorial of a number</TITLE>
   </HEAD>
   <BODY>
   <SCRIPT type = “text/javascript”>
    var a=5,fact=1;
    for(i=1;i<=5,i++)
    {
     fact=fact*i;
    }
    document.write(“factorial is”+fact);
</SCRIPT>
</BODY>
</HTML>
```

Non Primitive Data Type

- These **data types** are not actually defined by the programming language but are created by the programmer. They are also called “reference variables” or “object references” since they reference a memory location which stores the **data**. **Objects** such as functions and arrays are referred to as non-primitive values.
- The non-primitive data types are as follows:

Data Type	Description
Object	represents instance through which we can access members
Array	represents group of similar values
Functions	a set of statements that performs a task or calculates a value

Simple Example of Non Primitive Data Type

```
<html>
  <head>
  <title>
  </title>
  <script type="text/javascript">
   function hello()
   {
    document.write("hi/hello");
   }
  </script>
  <body>
  <script type="text/javascript">
  hello();
  </script>
  </body>
</html>
```

Javascript Operators

JavaScript operators are symbols that are used to perform operations on operands. For example:

var sum=10+20;

Here, + is the arithmetic operator and = is the assignment operator.

There are following types of operators in JavaScript.

- Arithmetic Operators
- Comparison (Relational) Operators
- Bitwise Operators
- Logical Operators
- Assignment Operators
- Special Operators

Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations on the operands. The following operators are known as JavaScript arithmetic operators.

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

```
<html>
<body>
 <script type = "text/javascript">
    var a = 33;
    var b = 10;
    var c = "Test";
    var linebreak = "<br />";
    document.write("a + b = ");
    result = a + b;
    document.write(result);
    document.write(linebreak);
    document.write("a - b = ");
    result = a - b;
    document.write(result);
    document.write(linebreak);
    document.write("a / b = ");
    result = a / b;
    document.write(result);
    document.write(linebreak);
    document.write("a % b = ");
    result = a % b;
    document.write(result);
    document.write(linebreak);
    a = ++a;
    document.write("++a = ");
    result = ++a;
    document.write(result);
    document.write(linebreak);
    b = --b;
    document.write("--b = ");
    result = --b;
    document.write(result);
    document.write(linebreak);
 </script>
</body>
</html>
```

14%3= ?

14%-3=?

-14%3=-?

-14%-4=?

14%3= 2

14%-3=2

-14%3=-2

-14%-4=-2

Comparison Operators

The JavaScript comparison operator compares the two operands. The comparison operators are as follows:

Operator	Description	Example
==	Is equal to	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

```
<html>
 <body>
  <script type = "text/javascript">
    var a = 10;
    var b = 20;
    var linebreak = "<br />";
    document.write("(a == b) => ");
    result = (a == b);
    document.write(result);
    document.write(linebreak);
    document.write("(a < b) => ");
    result = (a < b);
    document.write(result);
    document.write(linebreak);
    document.write("(a > b) => ");
    result = (a > b);
    document.write(result);
    document.write(linebreak);
    document.write("(a != b) => ");
    result = (a != b);
    document.write(result);
    document.write(linebreak);
    document.write("(a >= b) => ");
    result = (a >= b);
    document.write(result);
    document.write(linebreak);
    document.write("(a <= b) => ");
    result = (a <= b);
    document.write(result);
    document.write(linebreak);
  </script>
 </body>
</html>
```

Output :

(a == b) => false
(a < b) => true
(a > b) => false
(a != b) => true
(a >= b) => false
a <= b) => true

Logical Operators

The following operators are known as JavaScript logical operators.

Operator	Description	Example
&&	Logical AND	(10==20 && 20==33) = false
||	Logical OR	(10==20 || 20==33) = false
!	Logical Not	!(10==20) = true

```
<html>
  <body>
     <script type = "text/javascript">
        var a = true;
        var b = false;
        var linebreak = "<br />";
        document.write("(a && b) => ");
        result = (a && b);
        document.write(result);
        document.write(linebreak);
        document.write("(a || b) => ");
        result = (a || b);
        document.write(result);
        document.write(linebreak);
        document.write("!(a && b) => ");
        result = (!(a && b));
        document.write(result);
        document.write(linebreak);
     </script>
  </body>
</html>
```

Output :

(a && b) => false

(a || b) => true

!(a && b) => true

Assignment Operators

The following operators are known as JavaScript assignment operators.

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

```
<html>
  <body>
    <script type = "text/javascript">
      var a = 33;
      var b = 10;
      var linebreak = "<br />";
      document.write("Value of a => (a = b) => ");
      result = (a = b);
      document.write(result);
      document.write(linebreak);
      document.write("Value of a => (a += b) => ");
      result = (a += b);
      document.write(result);
      document.write(linebreak);
      document.write("Value of a => (a -= b) => ");
      result = (a -= b);
     document.write(result);
     document.write(linebreak);
     document.write("Value of a => (a *= b) => ");
     result = (a *= b);
     document.write(result);
     document.write(linebreak);
     document.write("Value of a => (a /= b) => ");
     result = (a /= b);
     document.write(result);
     document.write(linebreak);
     document.write("Value of a => (a %= b) => ");
     result = (a %= b);
     document.write(result);
     document.write(linebreak);
    </script>
  </body>
</html>
```

Output

```
Value of a => (a = b) => 10
Value of a => (a += b) => 20
Value of a => (a -= b) => 10
Value of a => (a *= b) => 100
Value of a => (a /= b) => 10
Value of a => (a %= b) => 0
```

Bitwise Operators

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

Operator	Description	Example
&	Bitwise AND	(10==20 & 20==33) = false
|	Bitwise OR	(10==20 | 20==33) = false
^	Bitwise XOR	(10==20 ^ 20==33) = false
~	Bitwise NOT	(~10) = -10
<<	Bitwise Left Shift	(10<<2) = 40
>>	Bitwise Right Shift	(10>>2) = 2

```
<html>
   <body>
      <script type = "text/javascript">
         var a = 2; // Bit presentation 10
         var b = 3; // Bit presentation 11
         var linebreak = "<br />";
         document.write("(a & b) => ");
         result = (a & b);
         document.write(result);
         document.write(linebreak);
         document.write("(a | b) => ");
         result = (a | b);
         document.write(result);
         document.write(linebreak);
         document.write("(a ^ b) => ");
         result = (a ^ b);
         document.write(result);
         document.write(linebreak);
         document.write("(~b) => ");
         result = (~b);
         document.write(result);
         document.write(linebreak);
         document.write("(a << b) => ");
         result = (a << b);
         document.write(result);
         document.write(linebreak);
         document.write("(a >> b) => ");
         result = (a >> b);
         document.write(result);
         document.write(linebreak);
      </script>
   </body>
</html>
```

Output :

```
(a & b) => 2
(a | b) => 3
(a ^ b) => 1
(~b) => -4
(a << b) => 16
(a >> b) => 0
```

Special Operators

The following operators are known as JavaScript special operators.

Operator	Description
(?:)	Conditional Operator returns value based on the condition. It is like if-else.
,	Comma Operator allows multiple expressions to be evaluated as single statement.
delete	Delete Operator deletes a property from the object.
in	In Operator checks if object has the given property
instanceof	checks if the object is an instance of given type
new	creates an instance (object)
typeof	checks the type of object.
void	it discards the expression's return value.

```
<html>
  <body>
     <script type = "text/javascript">
        var a = 10;
        var b = "String";
        var linebreak = "<br />";
        result = (typeof b == "string" ? "B is String" : "B is Numeric");
        document.write("Result => ");
        document.write(result);
        document.write(linebreak);
        result = (typeof a == "string" ? "A is String" : "A is Numeric");
        document.write("Result => ");
        document.write(result);
        document.write(linebreak);
     </script>
  </body>
</html>
```

Output

Result => B is String

Result => A is Numeric

Taking Run-Time Input

```
<html>
 <head>
  <title> runtime input</title>
 </head>
 <body>
  <script type="text/javascript">
   var a= parseInt(prompt("Enter the first no"));
   var b= parseInt(prompt("Enter the second no"));
   var c=a+b;
   document.write("The sum is: ",c);
  </script>
 </body>
</html>
```

Output:

Prompt boxes will open up and will ask for the inputs

JavaScript - if...else Statement

- **If Statement :** The **if** statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.

Syntax

The syntax for a basic if statement is as follows –

```
if (expression)
{
 Statement(s) to be executed if expression is true
}
```

```
<html>
 <body>
  <script type = "text/javascript">
   var age = parseInt(prompt("Enter the age"));
   if( age > 18 )
   {
    document.write("<b>Qualifies for Voting</b>");
   }
  </script>
 </body>
</html>
```

- **if...else statement :** The **'if...else'** statement is the next form of control statement that allows JavaScript to execute statements in a more controlled way.

```
if (expression)
{
 Statement(s) to be executed if expression is true
}
else
{
 Statement(s) to be executed if expression is false
}
```

```
<html>
 <body>
  <script type = "text/javascript">
   var age = parseInt(prompt("Enter the age"));
   if( age > 18 )
   {
    document.write("<b>Qualifies for driving</b>");
   }
   else
   {
    document.write("<b>Does not qualify for driving</b>");
   }
  </script>
 </body>
</html>
```

- **if...else if... statement**

It allows JavaScript to make a correct decision out of several conditions.

```
if (expression 1)
{
 Statement(s) to be executed if
 expression 1 is true
}
else if (expression 2)
 {
 Statement(s) to be executed if
expression 2 is true
 }
else if (expression 3)
{
 Statement(s) to be executed if
expression 3 is true
 }
else
{
 Statement(s) to be executed if no
expression is true
}
```

```
<html>
 <body>
  <script type = "text/javascript">
   var book = prompt("Enter the book name";
   if( book == "history" )
   {
    document.write("<b>History Book</b>");
   }
   else if( book == "maths" )
   {
    document.write("<b>Maths Book</b>");
   }
   else if( book == "economics" )
   {
    document.write("<b>Economics
Book</b>");
   }
   else
   {
    document.write("<b>Unknown Book</b>");
   }
  </script>
  </body>
 <html>
```

- Switch Case

The objective of a **switch** statement is to give an expression to evaluate several different statements based on the value of the expression.

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

- While Loops

The purpose of a **while** loop is to execute a statement or code block repeatedly as long as an **expression** is true. Once the expression becomes **false,** the loop terminates.

Syntax :

```
while (expression)
{
 Statement(s) to be executed if expression is true
}
```

• The do...while Loop

The **do...while** loop is similar to the **while** loop except that the condition check happens at the end of the loop. This means that the loop will always be executed at least once, even if the condition is **false**.

Syntax

```
do
{
 Statement(s) to be executed;
} while (expression);
```

- **For Loop**

Syntax :

```
for (initialization; test condition; iteration statement)
{
 Statement(s) to be executed if test condition is true
}
```

```
<html>
  <body>
    <script type = "text/javascript">
     <var count;
      document.write("Starting Loop" + "<br />");
      for(count = 0; count < 10; count++)
      {
       document.write("Current Count : " + count );
       document.write("<br />");
      }
       document.write("Loop stopped!");
     </script>
  </body>
 </html>
```

```
Starting Loop
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Current Count : 5
Current Count : 6
Current Count : 7
Current Count : 8
Current Count : 9
Loop stopped!
```

Math objects in Javascript

The **JavaScript math** object provides several constants and methods to perform mathematical operation. Unlike date object, it doesn't have constructors.

Math Properties

Sr.No.	Property & Description
1	LN2 : Natural logarithm of 2, approximately 0.693.
2	LN10 :Natural logarithm of 10, approximately 2.302.
3	PI: Ratio of the circumference of a circle to its diameter, approximately 3.14159.
4	SQRT1_2 :Square root of 1/2; equivalently, 1 over the square root of 2, approximately 0.707.
5	SQRT2: Square root of 2, approximately 1.414.

- **Math LN2 :** It returns the natural logarithm of 2 which is approximately 0.693.

```
<script type = "text/javascript">
  var value1 = Math.LN2
  document.write("Property Value is : " +
  value1);
</script>
```

```
Output :
Property Value is :
0.6931471805599453
```

- **Math LN10 :** It returns the natural logarithm of 10 which is approximately 2.302.

```
<script type = "text/javascript">
  var value1 = Math.LN10
  document.write("Property Value is : " +
  value1);
</script>
```

```
Output :
Property Value is :
2.302585092994046
```

- **Math PI :** It returns the PI value ie. 3.14159

```
<script type = "text/javascript">
 var value1 = Math.PI
  document.write("Property Value is : " +
  value1);
</script>
```

```
Output :
Property Value is :
3.141592653589793
```

- Math SQRT1_2 : It returns the square root of 1/2; equivalently, 1 over the square root of 2 which is approximately 0.707.

```
<script type = "text/javascript">
 var value1= Math.SQRT1_2
 document.write("Property Value is : " +
 value1);
</script>
```

```
Output :
Property Value is :
0.707106781865476
```

- Math SQRT2 : It returns the square root of 2 which is approximately 1.414

```
<script type = "text/javascript">
 var value1 = Math.SQRT2
 document.write("Property Value is : " +
 value1);
</script>
```

```
Output :
Property Value is :
1.4142135623730951
```

Math Methods

Methods	Description
abs()	It returns the absolute value of the given number.
ceil()	It returns a smallest integer value, greater than or equal to the given number.
cos()	It returns the cosine of the given number.
floor()	It returns largest integer value, lower than or equal to the given number.
max()	It returns maximum value of the given numbers.
min()	It returns minimum value of the given numbers.
pow()	It returns value of base to the power of exponent.
round()	It returns closest integer value of the given number.
sign()	It returns the sign of the given number
sin()	It returns the sine of the given number.
sqrt()	It returns the square root of the given number
tan()	It returns the tangent of the given number.
trunc()	It returns an integer part of the given number.

- abs() : This method returns the absolute value of a number.

```
Math.abs( x ) ;
```

```
Math.abs(-15); =15
Math.abs(20); =20
```

- ceil() : This method returns the smallest integer greater than or equal to a number.

```
Math.ceil( x ) ;
```

```
Math.ceil(45.95);
Math.ceil(45.25);
```

- cos() : This method returns the cosine of a number.

```
Math.cos( x ) ;
```

- sin() : This method returns the sine of a number.

```
Math.sin( x ) ;
```

- tan() : This method returns the tan of a number.

```
Math.tan( x ) ;
```

- floor() : This method returns the largest integer less than or equal to a number.

```
Math.floor( x ) ;
```

- log(x) : This method returns the natural logarithm (base E) of a number.

```
Math.log( x ) ;
```

- Max() : This method returns the largest of one or more numbers.

```
Math.max(value1, value2, ... valueN ) ;
```

- Min() : This method returns the smallest of one or more numbers.

```
Math.min(value1, value2, ... valueN ) ;
```

- Pow() : This method returns the base to the exponent power, that is, $\mathbf{base}^{\mathbf{exponent}}$.

```
Math.pow(base, exponent ) ;
```

- round() : This method returns the value of a number rounded to the nearest integer.

```
Math.round( x ) ;
```

- sqrt() : This method returns the square root of a number.

```
Math.sqrt( x ) ;
```

- trunc() : It returns an integer part of the given number.

```
Math.trunc( x ) ;
```

Date objects in Javascript

The Date object is a data type built into the JavaScript language. Date objects are created with the **new Date( )**.

```
var today = new Date();
document.write(today);
```

Date Object Methods

Sr.No.	Method & Description
1	Date() Returns today's date and time
2	getDate() Returns the day of the month for the specified date according to local time.
3	getDay() Returns the day of the week for the specified date according to local time.
4	getFullYear() Returns the year of the specified date according to local time.
5	getHours() Returns the hour in the specified date according to local time.
6	getMilliseconds()Returns the milliseconds in the specified date according to local time.
7	getMinutes() Returns the minutes in the specified date according to local time.
8	getMonth() Returns the month in the specified date according to local time.
9	getSeconds() Returns the seconds in the specified date according to local time.
10	getTime() Returns the numeric value of the specified date as the number of milliseconds since January 1, 1970, 00:00:00 UTC.
11	toLocaleString() It converts the object to the string.

Javascript code to demonstrate Date Object

```
<script type="text/javascript">
   var today = new Date();
   var dateString = today.toLocaleString();
   var month = today.getMonth();
   var year = today.getYear();
   var time = today.getTime();
   var hours = today.getHours();
   var minute = today.getMinute();
   var second = today.getSeconds();
   var milliseconds= today.getmilliseconds();
   document.write(dateString + month + year + time + hours + minute + second +
   miiliseconds);
<script>
```

String Properties and Methods

- The JavaScript String object is a global object that is used to store strings. A string is a sequence of letters, numbers, special characters and arithmetic values or combination of all.

String Properties

Property	Description
length	Returns the length of a string.

```
<script>
var str="JavaScript";
document.write(str.length);
</script>
```

String Methods

Method	Description
charAt()	Returns the character at the specified index.
concat()	Joins two or more strings, and returns a new string.
indexOf()	Returns the index of the first occurrence of the specified value in a string.
lastIndexOf()	Returns the index of the last occurrence of the specified value in a string.
replace()	Replaces the occurrences of a string or pattern inside a string with another string, and return a new string without modifying the original string.
slice()	Extracts a portion of a string and returns it as a new string.
split()	Splits a string into an array of substrings.
substr()	Extracts the part of a string between the start index and a number of characters after it.
substring()	Extracts the part of a string between the start and end indexes.
toLowerCase()	Converts a string to lowercase letters.
toString()	Returns a string representing the specified object.
toUpperCase()	Converts a string to uppercase letters.
trim()	Removes whitespace from both ends of a string.

1) charAt(index) : Returns the character at the given index.

```
<script>
  var str="javascript";
  document.write(str.charAt(2));
</script>
```

Output:
v

2) concat(str) : This concatenates or joins two strings.

```
<script>
  var s1="javascript ";
  var s2="concat example";
  var s3=s1.concat(s2);
  document.write(s3);
</script>
```

Output:
javascript concat example

3) indexOf(str) : Returns the index position of the given string.

```
<script>
  var s1="javascript from javatpoint indexof";
  var n=s1.indexOf("from");
  document.write(n);
<script>
```

Output:
11

4) lastIndexOf(str): Returns the last index position of the given string.

```
<script>
var s1="javascript from javatpoint indexof";
var n=s1.lastIndexOf("java");
document.write(n);
</script>
```

Output:
16

5) toLowerCase() : Returns the given string in lowercase letters.

```
<script>
var s1="JavaScript toLowerCase Example";
var s2=s1.toLowerCase();
document.write(s2);
</script>
```

Output:
javascript tolowercase example

6) toUpperCase(): Returns the given string in uppercase letters.

```
<script>
var s1="JavaScript toUpperCase Example";
var s2=s1.toUpperCase();
document.write(s2);
</script>
```

Output:
JAVASCRIPT TOUPPERCASE EXAMPLE

7) slice(beginIndex, endIndex): Returns the parts of string from given beginIndex to endIndex. In slice() method, beginIndex is inclusive and endIndex is exclusive.

```
<script>
var s1="abcdefgh";
var s2=s1.slice(2,5);
document.write(s2);
</script>
```

Output:
cde

8) trim() :Removes leading and trailing whitespaces from the string.

```
<script>
var s1="     javascript trim     ";
var s2=s1.trim();
document.write(s2);
</script>
```

Output:
javascript trim

9) split() : Splits a string into an array of substrings.

```
<script>
var str="This is JavaTpoint website";
document.write(str.split(" ")); //splits the given string.
</script>
```

10) substr(startindex, no. of characters): Extracts the part of a string between the start index and a number of characters after it.

```
<script>
var str="javascript";
document.write(str.substr(1,6)); //substring of   the given string.
</script>
```

11)replace(string, stringtoreplacewith): Replaces the occurrences of a string or pattern inside a string with another string, and return a new string without modifying the original string.

```
<script>
•var str="organization";
•document.write(str.replace('z','s')); //substring of   the given string.
•</script>
```

Code to demonstrate String Properties and Methods

```
<script type="text/javascript">
   var str= "george";
   var len=str.length;
   document.write("Length of String is "+ len);
   var str1="Computer";
   document.write("Character at position 4 is" + str.charAt(4));
   document.write("Position of o is" + str.indexOf('o'));
   document.write("The result of adding two strings is " + str.concat(str1));
   document.write("The substring is "+str.subStr(1,4));
   document.write("The result in small letters is "+str.toLowerCase());
   document.write("The result in upper letters is "+str.toUpperCase());
</script>
```

Dialog Boxes

JavaScript supports three important types of dialog boxes.

- **Alert**
- **Confirm**
- **Prompt**

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.

Alert box gives only one button "OK" to select and proceed.

```
<html>
 <head>
   <script type = "text/javascript">
     function Warn()
     {
      alert ("This is a warning message!");
      document.write ("This is a warning message!");
     }
    </script>
 </head>
 <body>
  <p>Click the following button to see the result: </p>
  <form>
    <input type = "button" value = "Click Me" onclick = "Warn();" />
  </form>
 </body>
</html>
```

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. We can use a confirmation dialog box as follows.

```
<html>
<head>
<script type = "text/javascript">
function getConfirmation()
{
var retVal = confirm("Do you want to continue ?");
if( retVal == true )
 {
document.write ("User wants to continue!");
return true;
 }
 else
{
document.write ("User does not want to continue!");
return false;
}
 }
</script>
</head>
 <body>
<p>Click the following button to see the result: </p>
<form>
<input type = "button" value = "Click Me" onclick =
"getConfirmation();" />
 </form>
</body>
 </html>
```

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 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**.

```
<html>
<head>
<script type = "text/javascript">
 function getValue()
 {
  var retVal = prompt("Enter your name : ",
"your name here");
  document.write("You have entered : " +
retVal);
 }
 </script>
 </head>
<body>
<p>Click the following button to see the result:
</p>
<form>
<input type = "button" value = "Click Me"
onclick = "getValue();" />
</form>
</body>
</html>
```

Arrays in Javascript

- The **Array** object lets you store multiple values in a single variable. It stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

  Syntax :Use the following syntax to create an **Array** object −

```
var fruits = new Array( "apple", "orange", "mango" );
```

- The **Array** parameter is a list of strings or integers. When you specify a single numeric parameter with the Array constructor, you specify the initial length of the array.

  You can create array by simply assigning values as follows −

```
var fruits = [ "apple", "orange", "mango" ];
```

- You will use ordinal numbers to access and to set values inside an array as follows.

Array Property

Sr.No.	Property & Description
1	Length : Reflects the number of elements in an array.

1. Program to input the values in an array and then print those values.
2. Program to find the largest among the values
3. Program to find the smallest no.
4. Program to sort the values in ascending order.
5. Program to find the middle value of the array.

Array Methods

Sr.No	Method & Description
1	concat()Returns a new array comprised of this array joined with other array(s) and/or value(s).
2	indexOf()Returns the first (least) index of an element within the array equal to the specified value, or -1 if none is found.
3	join()Joins all elements of an array into a string.
4	lastIndexOf()Returns the last (greatest) index of an element within the array equal to the specified value, or -1 if none is found.
5	pop()Removes the last element from an array and returns that element.
6	push()Adds one or more elements to the end of an array and returns the new length of the array.
7	reverse()Reverses the order of the elements of an array -- the first becomes the last, and the last becomes the first.
8	shift()Removes the first element from an array and returns that element.
9	slice()Extracts a section of an array and returns a new array.
10	sort()Sorts the elements of an array
11	toString()Returns a string representing the array and its elements.

1.Concat() Method:

Syntax:

The syntax of concat() method is as follows −

```
array.concat(value1, value2, ..., valueN);
```

```
<html>
 <head>
  <title>JavaScript Array concat Method</title>
 </head>
  <body>
    <script type = "text/javascript">
      var alpha = ["a", "b", "c"];
      var numeric = [1, 2, 3];
      var alphaNumeric = alpha.concat(numeric);
      document.write("alphaNumeric : " + alphaNumeric );
    </script>
  </body>
</html>
```

```
Output:
alphaNumeric : a,b,c,1,2,3
```

2. indexOf() Method: JavaScript array **indexOf()** method returns the first index at which a given element can be found in the array, or -1 if it is not present.

Syntax:

The syntax of indexOf() method is as follows −

- **searchElement** − Element to locate in the array.
- **fromIndex** − The index at which to begin the search.

```
array.indexOf(searchElement[, fromIndex]);
```

```
<html>
 <head>
  <title>JavaScript Array concat Method</title>
 </head>
  <body>
    <script type = "text/javascript">
      var arr = [12,5, 8, 130,44];
      var index = arr.indexOf(8);
      document.write("Index is : " + index );
    </script>
  </body>
</html>
```

```
Output:
index is : 2
```

3. join() Method: Array **join()** method joins all the elements of an array into a string.

Syntax:

- **separator** – Specifies a string to separate each element of the array. If omitted, the array elements are separated with a comma.

```
array.join(separator);
```

```
<script type = "text/javascript">
  var arr = new Array("First","Second","Third");
  var str = arr.join();
  document.write("str : " + str );
  var str = arr.join(":");
  document.write("<br />str : " + str );
  var str = arr.join(" + ");
  document.write("<br />str : " + str );
</script>
```

str : First,Second,Third
str : First: Second: Third
str : First + Second + Third

4. reverse() Method: The JavaScript array reverse() method changes the sequence of elements of the given array and returns the reverse sequence. In other words, the arrays last element becomes first and the first element becomes the last.

Syntax: `array.reverse();`

```
<script>
var arr=["Deepal", "Ravi", "Anmol"];
var rev=arr.reverse();
document.writeln(rev);
</script>
```

Output:
Anmol, Ravi, Deepak

5. lastIndexOf() Method: The JavaScript array lastIndexOf() method is used to search the position of a particular element in a given array. It behaves similar to indexOf() method with a difference that it start searching an element from the last position of an array

Syntax: array.lastIndexOf(element,index); element represents the element to be searched and **index** represents the index position from where search starts. It is optional.

```
<script>
var arr=["C","C++","Python","C++","Java"];
var result= arr.lastIndexOf("C++");
document.writeln(result);
</script>
```

Output:
3

```
<script>
var arr=["C","C++","Python","C++","Java"];
var result= arr.lastIndexOf("C++",2);
document.writeln(result);
</script>
```

Output:
1

6. push() Method: The JavaScript array push() method adds one or more elements to the end of the given array. This method changes the length of the original array.

Syntax: array.push(element1,element2....elementn)

```
<script>
var arr=["Deepak", "Ravi"];
arr.push("Anmol");
document.writeln(arr);
</script>

Output:
Deepak, Ravi, Anmol
```

7. pop() Method: The JavaScript array pop() method removes the last element from the given array and return that element. This method changes the length of the original array.

Syntax:

```
array.pop()
```

```
<script>
var arr=["Deepak", "Ravi", "Anmol"];
document.writeln("Orginal array: "+arr+"<br>");
document.writeln("Extracted element: "+arr.pop()+"<br>");
document.writeln("Remaining elements: "+ arr);
</script>
```

Output:

```
Orginal array: Deepak, Ravi, Anmol
Extracted element: Anmol
Remaining elements: Deepak, Ravi
```

8. shift() Method: The JavaScript array shift() method removes the first element of the given array and returns that element. This method changes the length of the original array.

Syntax:

```
array.shift()
```

```
<script>
var arr=["Deepak", "Ravi", "Anmol"];
var result=arr.shift();
document.writeln(result);
</script>
```

Output:

Deepak

9. slice() Method: The JavaScript array slice() method extracts the part of the given array and returns it. This method doesn't change the original array.

Syntax:

array.slice(start,end)

start - It is optional. It represents the index from where the method starts to extract the elements.

end - It is optional. It represents the index at where the method stops extracting elements.

```
<script>
var arr=["Deepak", "Ravi", "Anmol ", " Arvind "];
var result=arr.slice(1,2);
document.writeln(result);
</script>
```

Output:

Ravi

10. sort() Method: The JavaScript array sort() method is used to arrange the array elements in some order. By default, sort() method follows the ascending order.

Syntax: `array.sort()`

```
<script>
var arr=["Deepak", "Ravi", "Anmol ", " Arvind "];
var result=arr.sort();
document.writeln(result);
</script>
```

Output:

Anmol, Arvind, Deepak, Ravi

11. toString() Method: The toString() method is used for converting and representing an array into string form. It returns the string containing the specified array elements. Commas separate these elements, and the string does not affect the original array.

Syntax:

```
array.toString()
```

```
<script>
        var arr=['j','a','v','a','T','p','o','i','n','t']; //array elements
        var str=arr.toString(); //toString() method implementation
         document.write("After converting into string:  "+str);
</script>
```

Output:

After converting into string: j,a,v,a,T,p,o,i,n,t

JAVASCRIPT OBJECT HIERARCHY MODEL

- JavaScript window object represents a window that displays the document. All javascript objects are the properties of some objects. The properties of window object are viable to all the javascript scripts that appear in the window HTML document.
- All objects on a webpage are not created equally. Each exists in a set relationship with other objects in a webpage. The navigator occupies the topmost slot in the javascript object hierarchy model followed by window object and so on.
- The objects are organized in a hierarchy. This hierarchical structure applies to the organization of objects in a web document.

The objects are:

1) **Navigator:-** Examples of navigator includes Netscape navigator, google chrome, internet explorer, etc.

2) **Window:-** The window includes the document, location and the history.

3) **Document:-** The documents include the anchor tag (<a href>), link and form.

4) **Form:-** The form includes checkbox, radio buttons, textbox, textarea, submit, etc.

All objects have the following:

1. **Properties:** It determines the functionality of the objects.
2. **Methods:** It allows access to those property.
3. **Events:** It allows javascript to be connected to the object by being mapped to the appropriate javascript handler.

Example: Document object has the following:

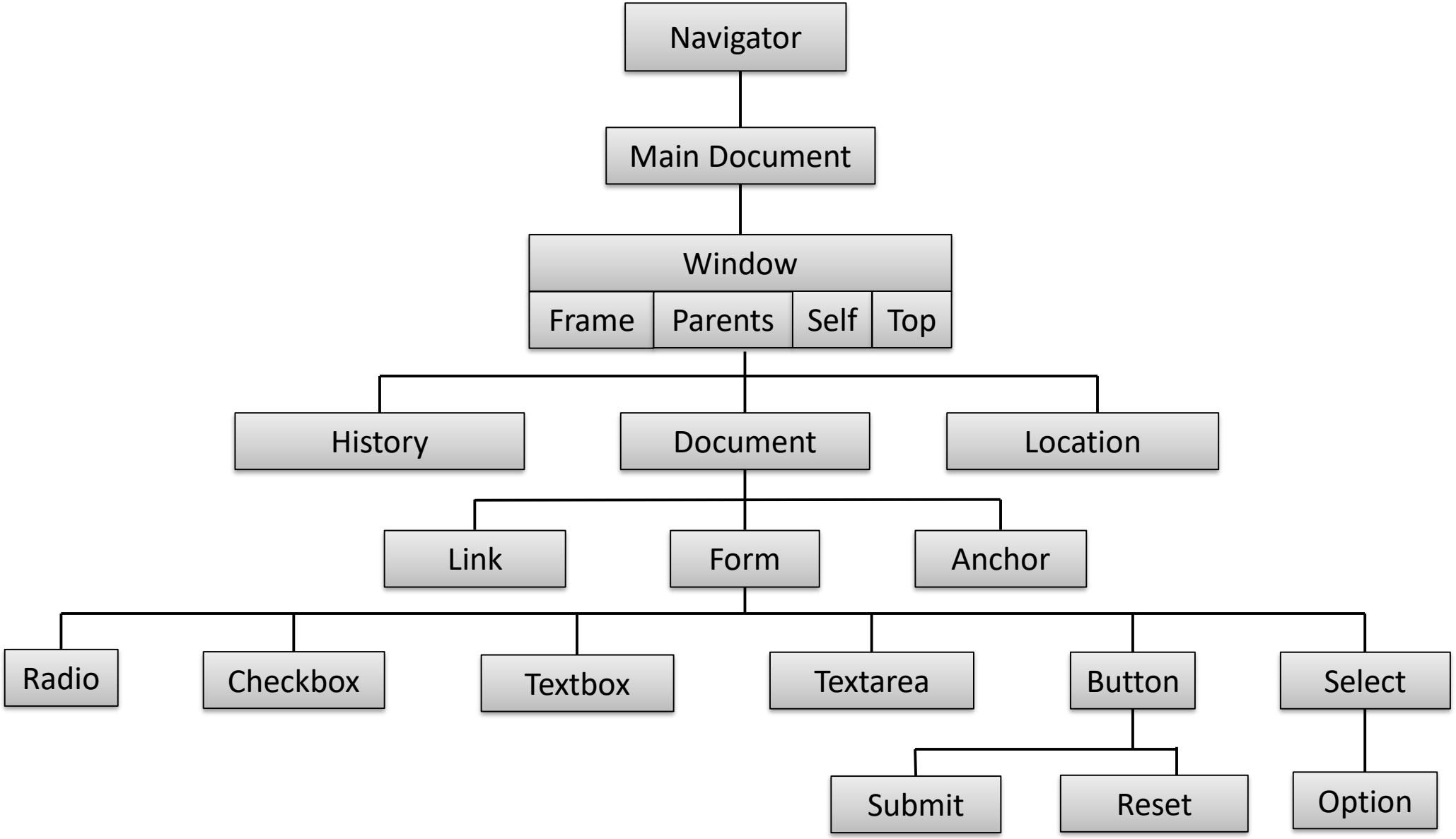
a) **Properties:** For e.g. BGCOLOR, BODY, TITLE, URL, etc.

b) **Methods:** For e.g. clear(), open(), write(), etc.

c) **Events:** For e.g. onClick(), onDblClick(), onMouseMove(), etc.

Similarly, the window objects will have the following:

a) **Properties:** Document, Event, Screen, Return values.
b) **Methods:** Alert, Confirm, Prompt, etc.
c) **Events:** onBlur, onFocus, etc.


Navigator
Main Document
Window
Frame
Parents
Self
Top
History
Document
Location
Link
Form
Anchor
Radio
Checkbox
Textbox
Textarea
Button
Select
Submit
Reset
Option

In the previous page, flowchart shows:

- Location specifies the URL;
- History specifies the URLs access within a window;
- The screen access the information about the color, size, etc. of the client computer screen.

HISTORY OBJECT

- The javascript history object represents an array of the URLs visited by the user.
- By using the object, we can load previous , forward (next) or any particular page.
- The history object is the window property so it can be accessed by window.history or simply history.

Properties/Methods of History Object	Use
1. Length(only property)	Indicates how many objects exists in that window.
2. Back()	Moves back in the history list i.e. loads the previous page.
3. Forward()	Moves forward in the history list i.e. loads the next page.
4. Go()	Goes to the URL in the history list i.e. loads the given page number.

```
<html>
 <head>
  <title> Using History Object </title>
  <script type="text/javascript">
   function goBack()
   {
     window.history.back(); //Moves back by one page
   }
   function goForward()
   {
     window.history.forward(); //Moves forward one page
   }
   function goBackTwo()
   {
     window.history.back(-2);  //Moves back by 2 pages
   }
   function goForwardTwo
   {
     window.history.forward(+2); //Moves forward by 2
                                 //pages
   }
  </script>
 </head>
 <body>
  <H1> Using History Objects </H1>
  <Form action=" ">
   <input type="button" value="<BackonePage"
          onClick="goBack()"/>
   <input type="button" value="<ForwardonePage"
          onClick="goForward()"/>
   <input type="button" value="<<Backtwopages"
          onClick="goBackTwo()"/>
   <input type="button" value=">>Forwardtwo
          pages" onClick="goForwardTwo()"/>
  </form>
 </body>
</html>
```

EVENT HANDLING IN JAVASCRIPT

- Javascript's interaction with HTML is handled through events that occur when user or the browser manipulates a page.
- An event is a notification that is something specific has occurred either within the browser such as the completion of loading of document or because of a browser user action such as a mouse click on a form button.
- When the page loads, it is called an event. When a user clicks a button, that click is also an event.
- An event handler is a script that is implicitly executed in response to the appearance to the event. Event handler enables a web document to be responsive to the browser and the user activities.

Event Tags and Attributes

Events are associated with HTML tag attribute which can be used to connect event to the handlers. The attributes have names that are closely related to their associated events.

onClick:- This is the most frequently use event which occurs when a user clicks the left button of his mouse.

```
<html>
 <head>
  <script type="text/javascript">
   function hello()
   {
     alert("Hello Everyone!");
   }
  </script>
 </head>
```

```
 <body>
  <form>
   <input type="button" onClick="Hello()"
          value="hello"/>
  </form>
 </body>
</html>
```