JAVASCRIPT

- Lightweight
- Interpreted
- Dynamic language for the web, integrated with HTML
- Can be used both for front end and backend
- No special environment setup required
- Can be used for client side validation
- Manipulating HTML pages
- Raise dynamic notifications and pop ups
- Javascript at client side
    - Eg. Validations, button clicks, navigations
- Less server interaction – helps validate user input before sending the page off to the server → less load on your server.
- Immediate feedback to the visitors – users need not wait for a page reload to see if they have forgotten to enter something.
- Increased interactivity – helps 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.

Javascript code is embedded in HTML for execution. Javascript tag to be used in HTML page :

```
<script language = "javascript" type = "text/javascript">
   JavaScript code
</script>
```

Other ways to include javascript in HTML : tag : **<script>... </script>**

- The <script> tag alerts the browser program to start interpreting all the text between these tags as a script.Script in <head>...</head> section.
- Script in <body>...</body> section.
- Script in <body>...</body> and <head>...</head> sections.
- Script in an external file and then include in <head>...</head> section. Extension of external JS file should be **.js**

Sample Code 1

```
<html>
   <body>
      <script language = "javascript" type = "text/javascript">
         <!--
            document.write("Welcome to first session of JS!")
         //-->
      </script>
   </body>
</html>
```

Some basic syntax :

- Space, tabs and newlines are ignored by the interpreter
- Use of semicolon is not required if each statement is placed on a separate line
- When using a single line for multiple comments, semicolon is necessary
- JS is case sensitive
- Any text between a // and the end of a line is treated as a comment and is ignored by JavaScript.
- Any text between the characters /* and */ is treated as a comment. This may span multiple lines.
- JavaScript also recognizes the HTML comment opening sequence <!--. JavaScript treats this as a single-line comment, just as it does the // comment.
- The HTML comment closing sequence --> is not recognized by JavaScript so it should be written as //-->.

How to manage JavaScript in Chrome

- Click the Chrome menu at the top right hand corner of your browser.
- Select **Settings**.
- Click **Show advanced settings** at the end of the page.
- Under the **Privacy** section, click the Content settings button.
- In the "Javascript" section, select "Do not allow any site to run JavaScript" or "Allow all sites to run JavaScript (recommended)".

Data Types in Javascript :

- **Numbers,** eg. 101,101.4
- **Strings** of text e.g. "Hello all"
- **Boolean** i.e. true or false.

Every variable has to be declared before its used.

Keyword for declaring a variable: **var**

JavaScript variable names should not start with a numeral (0-9). They must begin with a letter or an underscore character. JavaScript variable names are case-sensitive.

JavaScript is **untyped** language → a JavaScript variable can hold a value of any data type. You don't have to tell JavaScript during variable declaration what type of value the variable will hold. The value type of a variable can change during the execution of a program and JavaScript takes care of it automatically.

```
1. <script type = "text/javascript">
   <!--
      var a;
      var b;
   //-->
</script>


2. <script type = "text/javascript">
   <!--
      var a,b;
   //-->
</script>


3. <script type = "text/javascript">
   <!--
      var name = "ABC";
      var sum;
      sum = 2050;
   //-->
</script>
```

Scope of a variable

- **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.

Within the body of a function, a local variable takes precedence over a global variable with the same name. If you declare a local variable or function parameter with the same name as a global variable, you effectively hide the global variable.

JavaScript Reserved Words

A list of all the reserved words in JavaScript are given in the following table. They cannot be used as JavaScript variables, 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	

Operators in Javascript

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

Value of A=10 , B= 20

Sr.No.	Operator & Description
1	**+ (Addition)** Adds two operands **Eg:** A + B = 30
2	**- (Subtraction)** Subtracts the second operand from the first **Eg:** A – B = -10
3	*** (Multiplication)** Multiply both operands **Eg:** A * B = 200
4	**/ (Division)** Divide the numerator by the denominator **Eg:** B / A = 2
5	**% (Modulus)** Outputs the remainder of an integer division **Eg:** B % A = 0
6	**++ (Increment)** Increases an integer value by one **Eg:** A++ = 11

7	**-- (Decrement)** Decreases an integer value by one **Eg:** A-- = 9

Comparison Operators

A = 10 and B = 20

Sr.No.	Operator & Description
1	**= = (Equal)** Checks if the value of two operands are equal or not, if yes, then the condition becomes true. **Ex:** (A == B) is not true.
2	**!= (Not Equal)** Checks if the value of two operands are equal or not, if the values are not equal, then the condition becomes true. **Ex:** (A != B) is true.
3	**> (Greater than)** Checks if the value of the left operand is greater than the value of the right operand, if yes, then the condition becomes true. **Ex:** (A > B) is not true.
4	**< (Less than)** Checks if the value of the left operand is less than the value of the right operand, if yes, then the condition becomes true. **Ex:** (A < B) is true.
5	**>= (Greater than or Equal to)** Checks if the value of the left operand is greater than or equal to the value of the right operand, if yes, then the condition becomes true. **Ex:** (A >= B) is not true.
6	**<= (Less than or Equal to)** Checks if the value of the left operand is less than or equal to the value of the right operand, if yes, then the condition becomes true. **Ex:** (A <= B) is true.

Logical Operators

A = 10 and B = 20

Sr.No.	Operator & Description
1	**&& (Logical AND)** If both the operands are non-zero, then the condition becomes true. **Ex:** (A && B) is true.
2	**|| (Logical OR)** If any of the two operands are non-zero, then the condition becomes true. **Ex:** (A || B) is true.
3	**! (Logical NOT)** Reverses the logical state of its operand. If a condition is true, then the Logical NOT operator will make it false. **Ex:** ! (A && B) is false.

Bitwise Operators

A = 2 and B = 3

Sr.No.	Operator & Description
1	**& (Bitwise AND)** It performs a Boolean AND operation on each bit of its integer arguments. **Ex:** (A & B) is 2.
2	**| (BitWise OR)** It performs a Boolean OR operation on each bit of its integer arguments. **Ex:** (A | B) is 3.
3	**^ (Bitwise XOR)** It performs a Boolean exclusive OR operation on each bit of its integer arguments.

	Exclusive OR means that either operand one is true or operand two is true, but not both. **Ex:** (A ^ B) is 1.
4	**~ (Bitwise Not)** It is a unary operator and operates by reversing all the bits in the operand. **Ex:** (~B) is -4.
5	**<< (Left Shift)** It moves all the bits in its first operand to the left by the number of places specified in the second operand. New bits are filled with zeros. Shifting a value left by one position is equivalent to multiplying it by 2, shifting two positions is equivalent to multiplying by 4, and so on. **Ex:** (A << 1) is 4.
6	**>> (Right Shift)** Binary Right Shift Operator. The left operand's value is moved right by the number of bits specified by the right operand. **Ex:** (A >> 1) is 1.
7	**>>> (Right shift with Zero)** This operator is just like the >> operator, except that the bits shifted in on the left are always zero. **Ex:** (A >>> 1) is 1.

Assignment Operators

JavaScript supports the following assignment operators −

Sr.No.	Operator & Description
1	**= (Simple Assignment)** Assigns values from the right side operand to the left side operand **Ex:** C = A + B will assign the value of A + B into C
2	**+= (Add and Assignment)** It adds the right operand to the left operand and assigns the result to the left operand. **Ex:** C += A is equivalent to C = C + A

3	**−= (Subtract and Assignment)** It subtracts the right operand from the left operand and assigns the result to the left operand. **Ex:** C -= A is equivalent to C = C – A
4	***= (Multiply and Assignment)** It multiplies the right operand with the left operand and assigns the result to the left operand. **Ex:** C *= A is equivalent to C = C * A
5	**/= (Divide and Assignment)** It divides the left operand with the right operand and assigns the result to the left operand. **Ex:** C /= A is equivalent to C = C / A
6	**%= (Modules and Assignment)** It takes modulus using two operands and assigns the result to the left operand. **Ex:** C %= A is equivalent to C = C % A

Miscellaneous Operator

We will discuss two operators here that are quite useful in JavaScript: the **conditional operator** (? :) and the **typeof operator**.

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

Sr.No.	Operator and Description
1	**? : (Conditional)** If Condition is true? Then value X : Otherwise value Y

```
<html>
   <body>
      <script type = "text/javascript">
         <!--
            var a = 10;
            var b = 20;
            var linebreak = "<br />";
```

```
            document.write ("((a > b) ? 100 : 200) => ");
            result = (a > b) ? 100 : 200;
            document.write(result);
            document.write(linebreak);

            document.write ("((a < b) ? 100 : 200) => ");
            result = (a < b) ? 100 : 200;
            document.write(result);
            document.write(linebreak);
         //-->
      </script>
      <p>Set the variables to different values and different
operators and then try...</p>
   </body>
</html>
```

Output

```
((a > b) ? 100 : 200) => 200
((a < b) ? 100 : 200) => 100
Set the variables to different values and different operators and
then try...
```