



## Introduction to JavaScript

### Learning Objectives:

- To understand about JavaScript Language and advantages of JavaScript
- To understand the importance of Client side code, Steps to follow to JavaScript Programs.
- To understand JavaScript Variables, Declaring Variables and Rules for naming Variables – Scope of Variables and assigning values to variables.
- To develop the skills of Web page development using HTML and CSS
- To acquire the skills of web scripting using JavaScript
- To develop the skills of Internet and Online applications



### 14.1 Introduction to JavaScript:

On December 4, 1995, Netscape and Sun Inc. jointly introduced JavaScript 1.0. JavaScript had truly bridged the gap between the simple world of HTML and the more complex Common Gateway Interface (CGI) programs on the Server. It provides a common scripting language for Web developers to design, test and deploy Internet Applications.

The JavaScript client-side technology provides many advantages over traditional CGI Server-side scripts. For example, JavaScript code can be used to check if the user has entered a valid e-mail address in a form field. The JavaScript code is executed when the user click **Submit** button in the form, and only if all the entries are valid, they would be submitted to the Web Server.

### 14.2 Advantages of JavaScript Programming Language

- In HTML chapter we have learnt how to develop static web pages. But in real life web pages must be interactive. So to develop such interactive pages (Dynamic Web page ) JavaScript programming language is used.
- User entered data in the Dynamic Web page can be validated before sending it to the server. This saves server traffic, which means less load on your server.
- JavaScript includes such items as Textboxes, Buttons, drag-and-drop components and sliders to give a Rich Interface to site visitors. For example Creating a New email account in any service provider.

### 14.3 Using JavaScript in HTML page with <script> tag :

JavaScript can be implemented using <script>... </script> tags. The <script> tag containing JavaScript can be placed anywhere within in the web page, but it is normally recommended that should be kept it within the <head> tags. The <script> tag alerts the

browser program to start interpreting all the text between these tags as a script commands.

The syntax of JavaScript segment in Hyper Text Markup Language (HTML) or Dynamic Hyper Text Markup Language (DHTML) is as follows:

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

JavaScript code

```
</script>
```

The <SCRIPT> tag takes two important attributes –

**Language** – This attribute specifies that the scripting language. Typically, its value will be **javascript**. Although recent versions of HTML (EXtensible HyperText Markup Language - XHTML, its successor) have phased out the use of this attribute is optional.

**Type** – This attribute is used to indicate the scripting language and its value should be set to "text/javascript".

#### 14.3.1 Steps to follow to code JavaScript Language

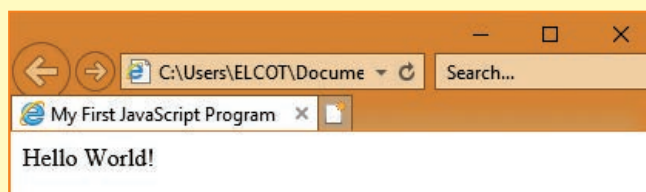
- Enter HTML and JavaScript code using any text editor.
- Save the latest version of this code.
- Use any browser to see the result. For example : Internet Explorer, Google Chrome, etc.,
- If this is a new document, open the file via browser's **Open Menu**. If the document is already loaded in the Memory, to reload the file into the browser use "**Refresh**" or press **F5** button.

#### 14.3.2 First JavaScript Program

##### Illustration 14.1 Simple JavaScript Program

```
<Html>
  <Head>
    <Title>My First JavaScript Program</Title>
    <script language="javascript" type="text/javascript">
      document.write("Hello World!")
    </script>
  </Head>
  <Body>
  </Body>
</Html>
```

**Output:**



#### 14.4 Lexical Structure of a JavaScript Program

The lexical structure of a programming language is the set of elementary rules that specifies how to write programs in that language. It is the lowest-level syntax of a language. The



Lexical structure specifies variable names, the delimiter characters for comments, and how one program statement is separated from the next.

- Though JavaScript is a case-sensitive language. It is good programming practice to type the command in lowercase.
- JavaScript ignores spaces that appear between tokens (identifiers, operators, punctuator, constants and keywords) in programs.
- JavaScript supports two styles of comments. Any text follow a “//” and the end of a line is treated as a single line comment and is ignored by JavaScript. Any text between the characters “/\* \*/” is also treated as a multiline comment.
- JavaScript uses the semicolon (;) to separate statements. Many JavaScript programmers use semicolons to explicitly mark the ends of statements.
- A literal is a data value for variable that appears directly in a program.
- An identifier is simply a name. In JavaScript, identifiers are used to name variables, functions and to provide labels for certain loops in JavaScript code.
- In JavaScript certain **keywords** are used as reserved words, These words cannot used as identifiers in the programs

### 14.5 JavaScript Variables:

Variable is a memory location where value can be stored. Variable is a symbolic name for a value. Variables are declared with the **var** keyword in JavaScript. Every variable has a name, called identifier.

#### 14.5.1 Basic Data types and Declaring variables:

Every variable has a data type that indicates what kind of data the variable holds. The basic data types in JavaScript are Strings, Numbers, and Booleans.

- A **string** is a list of characters, and a string literal is indicated by enclosing the characters in single or double quotes. Strings may contain a single character or multiple characters, including whitespace and special characters such as **\n** (the newline).
- **Numbers** can be integer or floating-point numerical value and numeric literals are specified in the natural way.
- **Boolean** can be any one of two values: **true** or **false**. Boolean literals are indicated by using true or false directly in the source code.

Variables are declared in JavaScript using var keyword that allocates storage space for new data and indicates to the interpreter that a new identifier is in use. Declaring a variable in JavaScript as follows:

```
var no;  
var no1,no2;
```

The **var no;** statement tells the interpreter that a new variable **no** is about to be used and **var no1,no2;** tells the interpreter that **no1** and **no2** are variables.

#### 14.5.2 Rules for naming variable

1. The first character must be a letter or an underscore (\_). Number cannot be as the first



character.

2. The rest of the variable name can include any letter, any number, or the underscore. You can't use any other characters, including spaces, symbols, and punctuation marks.
3. JavaScript variable names are case sensitive. That is, a variable named **RegisterNumber** is treated as an entirely different variable than one named **registernumber**.
4. There is no limit to the length of the variable name.
5. JavaScript's reserved words cannot be used as a variable name. All programming languages have a supply of words that are used internally by the language and that cannot be used for variable names.

### 14.5.3 Scope of variables

The scope of a variable is the life time of a variable of source code in which it is defined.

- A global variable has global scope; it can be defined everywhere in the JavaScript code.
- Variables declared within a function are defined only within the body of the function. They are local variables and have local scope.

### 14.5.4 Assigning values to variables

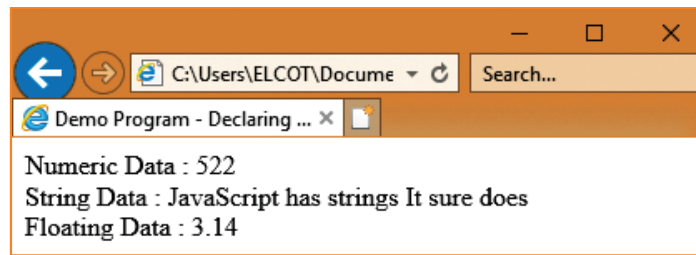
Variables can be assigned initial values when they are declared as follows:

```
var numericData1 = 522;  
var stringData = "JavaScript has strings\n It sure does";  
var numericData = 3.14;  
var booleanData = true;
```

#### Illustration 14.2 Declaring Variables

```
<Html>  
<Head>  
  <Title>Demo Program - Declaring Variables in JavaScript </Title>  
</Head>  
<Body>  
  <script language="javascript" type="text/javascript">  
    var numericData1 = 522;  
    var stringData = " JavaScript has strings\n It sure does";  
    var numericData = 3.14;  
    var booleanData = true;  
    document.write("Numeric Data : "+numericData1);  
    document.write("<br> String Data : "+stringData);  
    document.write("<br> Floating Data : "+numericData);  
  </script>  
</Body>  
</Html>
```

## Output



In addition, multiple variables can be declared with one **var** statement, if the variables are separated by commas:

```
var no1=50, no2=5065;
```

JavaScript allows the implicit declaration of variables by using them on the left-hand side of an assignment. In JavaScript there is no need to indicate data type during variable declarations. JavaScript variables are untyped and it is dynamically datatyped which means initially you can assign a value of any data type to a variable and later you can assign a value of different data type to the same variable. For example:

```
var value=100;
```

```
var value="JavaScript";
```

### 14.5.5 JavaScript Literals

A literal is a fixed value given to a variable in source code. Literals are often used to initialize variables. Values may be Integer, Floating point, Character, String and Boolean. For Example,

```
var int_const=250; //Integer constant//
```

```
var float_const=250.85; //Floating point constant//
```

```
var char_const='A'; //Character constant//
```

```
var string_const="Raman"; //String constant//
```

```
var boolean_const=true; //Boolean constant//
```

### write statement:

#### General Syntax:

```
document.write ("string " + var);
```

### 14.5.6 Type casting in JavaScript.

Type conversion is the act of converting one data type into a different data type which is also called as casting. In JavaScript there are two type of **casting**,

- Implicit casting and
- Explicit casting

Implicit casting occurs automatically in JavaScript when you change the data stored in a variable:



## 14.6 JavaScript Operators and Expressions

An operator combines the values of its operands in some way and evaluates to a new value. Operators are used for JavaScript's arithmetic expressions, comparison expressions, logical expressions, assignment expressions.

An expression is a phrase of JavaScript that a JavaScript interpreter can evaluate to produce a value. The data types are used directly as literals or within variables in combination with simple operators, such as addition, subtraction, and so on, to create an expressions. An expression is a code fragment that can be evaluated to some data type the language supports. An expression is simply one or more variables and/or constants joined by operators. An expression is evaluated and produces a result. The result of all expressions may be either an integer or floating-point value or Boolean value. There are three types of expressions as follows,

- Arithmetic expressions
- Relational expressions
- Logical expressions

### 14.6.1 Arithmetic Operators

JavaScript supports all the basic arithmetic operators like addition (+), subtraction (-), multiplication (\*), division (/), and modulus (%), also known as the remainder operator).

Table: 14.1 – Arithmetic Operators

Arithmetic Operator	Meaning	Example	Result
+	Addition	var sum = 20 + 120	Variable sum = 140
-	Subtraction	var diff = 20 - 120	Variable diff = 100
*	Multiplication	var prod = 10 * 100	Variable prod = 1000
/	Division	var res = 100/522	Variable res = 5.22
%	Modulus operator	var rem = 100 % 522	Variable rem = 22 (remainder)

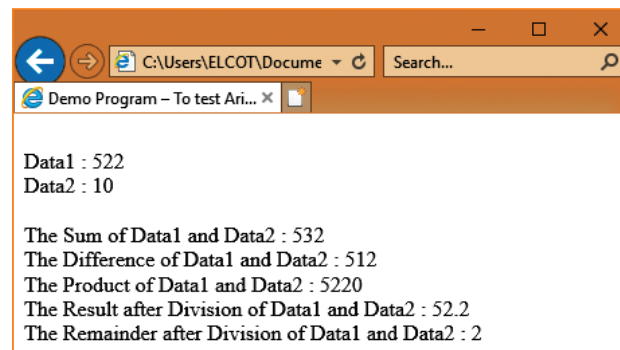
### Illustration 14.3 Using Arithmetic Operators

```
<Html>
<Head>
    <Title>Demo Program – To test Arithmetic Operators in JavaScript
</Title>
</Head>
<Body>
    <script language="javascript" type="text/javascript">
```



```
var value1 = 522, value2=10;
document.write("<br>Data1 : "+value1);
document.write("<br>Data2 : "+value2);
var sum = value1+value2;
var diff = value1-value2;
var prod = value1*value2;
var res = value1/value2;
var rem = value1%value2;
document.write("<br><br>The Sum of Data1 and Data2 : "+sum);
document.write("<br>The Difference of Data1 and Data2 : "+diff);
document.write("<br>The Product of Data1 and Data2 : "+prod);
document.write("<br>The Result after Division of Data1 and Data2 : "+res);
document.write("<br>The Remainder after Division of Data1 and Data2 : "+rem);
</script>
</Body>
</Html>
```

### Output:



### 14.6.2 Assignment Operator

An assignment operator is the operator used to assign a new value to a variable. Assignment operator can also be used for logical operations such as bitwise logical operations or operations on integral operands and Boolean operands.

In JavaScript = is an assignment operator, which is used to assign a value to a variable. Often this operator is used to set a variable to a literal value, for example,

```
var number1=10;
var number2=number1;
var name="Computer Science";
var booleanvar=true;
```

The assignment operator is used to assign a value to a single variable, but it is possible to perform multiple assignments at once by stringing them together with the = operator. For example, the statement



**var m = n = z = 25; // sets all three variables to a value of 25//**

The assignment operator can also be used to set a variable to hold the value of an expression. For example,

**var x = 102 + 5 - 50; // x set to 57 //**

JavaScript supports some shorthand arithmetic operators like +=, -=, \*=, /= and %= to evaluate arithmetic calculations.

Table: 14.2 Shorthand Arithmetic operators

Shorthand Arithmetic Operator	Meaning	Example	Result
+=	Add and assign	var sum = 120; sum += 20;	Variable sum = 140
-=	Subtract and assign	var diff = 120; diff -= 20;	Variable diff = 100
*=	Multiply and assign	var prod = 100; prod *=10;	Variable prod = 1000
/=	Division	Var res = 522; Res/=100	Variable res = 5.22
%=	Modulus operator	Var rem = 522; rem %= 100	Variable rem = 22 (remainder)

#### Illustration 14.4 Using Arithmetic Shorthand Operators

```
<Html>
  <Head>
    <Title>Demo Program - To test Arithmetic Shorthand Operators in JavaScript
  </Title>
  </Head>
</Html>
<Head>
  <Title>Demo Program - To test Arithmetic Shorthand Operators in JavaScript
</Title>
</Head>
<Body>
  <script language="javascript" type="text/javascript">
    var value1 = 522, value2=10;
    document.write("<br>Data1 : "+value1);
    document.write("<br>Data2 : "+value2);
    var sum = value1; sum+=value2;
    var diff = value1; diff-=value2;
    var prod = value1; prod*=value2;
    var res = value1; res/=value2;
```

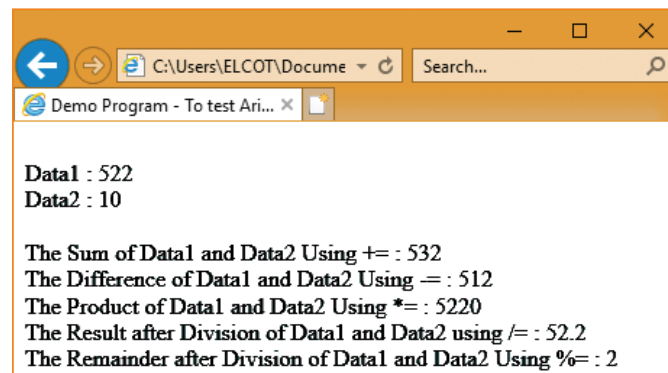


```

var rem = value1; rem%=value2;
document.write("<br><br>The Sum of Data1 and Data2 Using += : "+sum);
document.write("<br>The Difference of Data1 and Data2 Using -= : "+diff);
document.write("<br>The Product of Data1 and Data2 Using *= : "+prod);
document.write("<br>The Result after Division of Data1 and Data2 using /= : "+res);
document.write("<br>The Remainder after Division of Data1 and Data2 Using %= : "+rem);
</script>
</Body>
</Html>

```

## Output:



## Assignment:

### Develop JavaScript code for the following:

1. To find Simple Interest for the given Principle, Number of years and Rate of interest.
2. To find Compund Interest for the given Principle, Number of years and Rate of interest.
3. To find difference between Simple Interest and Compound Interst.

### 14.6.3 Relational or Comparison Operators:

Relational operators are also called as Comparison operators, they compares two values and the result is true or false. JavaScript provides a rich set of relational operators including == (equal to), != (not equal to), < (less than), > (greater than), <= (less than or equal to), and >= (greater than or equal to). Using a relational operator in an expression causes the expression to evaluate as true if the condition holds or false if otherwise.

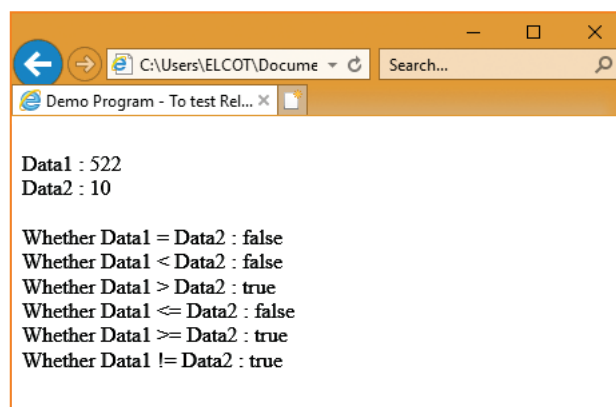
Table: 14.3 Relational or Comparison operators

Relational (Comparison) Operator	Meaning	Example	Result
Assume x=10 and y=20			
==	Equality	x==y	False
!=	In-equality	x!=y	True
<	Less-than	x<y	True
>	Greater-than	x>y	False
<=	Less-than or equal to	x<=y	True
>=	Greater-than or equal to	x>=y	False

### Illustration 14.5 Using Relational Operators

```
<Html>      <Head>
<Title>Demo Program - To test Relational(Comparison) Operators in JavaScript </Title>
</Head>      <Body>
    <script language="javascript" type="text/javascript">
        var value1 = 522, value2=10;
        document.write("<br>Data1 : "+value1);
        document.write("<br>Data2 : "+value2);
        document.write("<br><br>Whether Data1 = Data2 : "+(value1==value2));
        document.write("<br>Whether Data1 < Data2 : "+(value1<value2));
        document.write("<br>Whether Data1 > Data2 : "+(value1>value2));
        document.write("<br>Whether Data1 <= Data2 : "+(value1<=value2));
        document.write("<br>Whether Data1 >= Data2 : "+(value1>=value2));
        document.write("<br>Whether Data1 != Data2 : "+(value1!=value2)); </script>
    </Body> </Html>
```

#### Output:



#### 14.6.4 Logical Operators:

Logical operators perform logical (boolean) operations. Logical operators combine or invert boolean values. Once comparisons are made, the logical operators && (AND), || (OR) and ! (NOT) can be used to create more complex conditions.

Table: 14.4 Logical or Boolean operators

Logical Operator	Example	Meaning	Result
&&	((4<5) && (10>5)) ((expr1) && (expr2))	(Logical AND) Returns true if expr1 and expr2 both true.	True
	((4<5)    (10>5)) ((expr1)    (expr2))	(Logical OR) Returns true if either expr1 or expr2 is true, or both are true.	True
!	!(10>5) !(expr1)	(Logical NOT) Returns true if expr1 is false; otherwise, returns false.	False

### Usage :

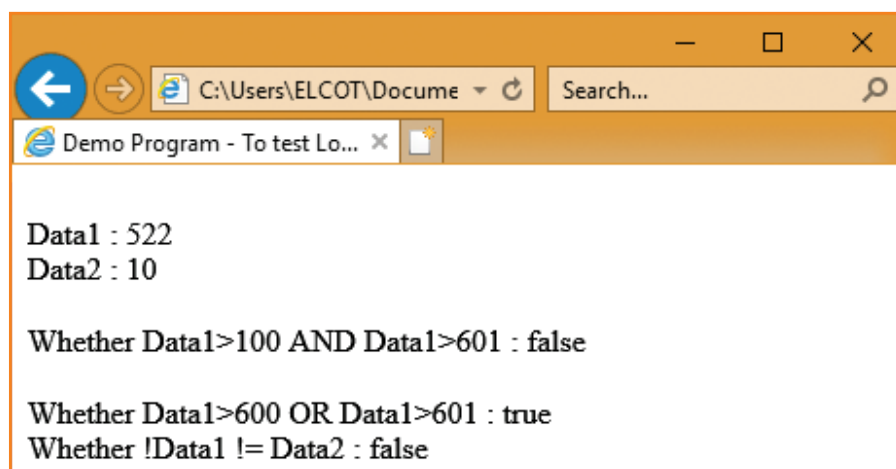
Best practice is to use logical operators on boolean operands. However, operands of any type can be combined. The strict rules are as follows:

- For && (AND) the result is false if the first operand is false; otherwise, the result is the Boolean value of the second operand.
- For || (OR) the result is true if the first operand is true; otherwise, the result is the Boolean value of the second operand.
- For ! (NOT) the result is true if the operand is false; otherwise, the result is false.

#### Illustration 14.6 Using Logical Operators

```
<Html>
<Head>
  <Title>Demo Program - To test Logical Operators in JavaScript </Title>
</Head>
<Body>
<script language="javascript" type="text/javascript">
  var value1 = 522, value2=10;
  document.write("<br>Data1 : "+value1);
  document.write("<br>Data2 : "+value2);
  var res1=((value1>100) && (value1>601));
  var res2=((value1>100) || (value1>601));
  var res3=(!(value1!=value2));
  document.write("<br><br>Whether Data1>100 AND Data1>601 : "+res1);
  document.write("<br><br>Whether Data1>600 OR Data1>601 : "+res2);
  document.write("<br>Whether !Data1 != Data2 : "+res3);
</script>
</Body>
</Html>
```

### Output:



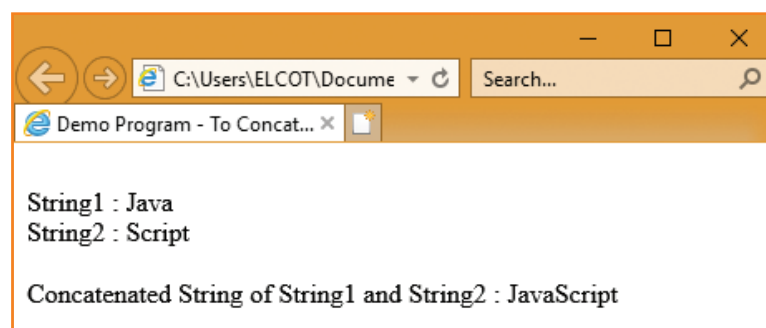
### 14.6.5 String Operators:

One of the built-in features of JavaScript is the ability to concatenate strings. The + operator performs addition on numbers but also serves as the concatenation operator for strings. Because string concatenation has precedence over numeric addition, + will be interpreted as string concatenation if any of the operands are strings. + operator which is also called as the string concatenation operator. For example:

#### Illustration 14.7 Using + Operator for concatenating String

```
<Html>
<Head>
  <Title>Demo Program - To Concatenating (+) Operators in JavaScript </Title>
</Head>
<Body>
  <script language="javascript" type="text/javascript">
    var String1 = "Java";
    var String2 = "Script";
    var String3=String1+String2;
    document.write("<br>String1 : "+String1);
    document.write("<br>String2 : "+String2);
    document.write("<br><br>Concatenated String of String1 and String2 : "+String3);
  </script>
</Body>
</Html>
```

#### Output:



### 14.6.6 Increment and Decrement Operators:

The ++ operator increments its single operand. The operator converts its operand to a number, adds 1 to that number, and assigns the incremented value back into the variable. The return value of the ++ operator depends on its position relative to the operand. When ++ is used before the operand, where it is known as the pre-increment operator, it increments the operand and evaluates to the incremented value of that operand. When used after the operand, where it is known as the post-increment operator, it increments its operand but evaluates to the un-incremented value of that operand. Consider the difference between these two lines of code:



**var m = 1, n = ++m; // m and n are both 2**

**var m = 1, n = m++; // m is 2, n is 1**

The -- operator decrements its single operand. It converts the value of the operand to a number, subtracts 1, and assigns the decremented value back to the operand. Like the ++ operator, the return value of -- depends on its position relative to the operand. When used before the operand, it decrements and returns the decremented value. When used after the operand, it decrements the operand but returns the undecremented value.

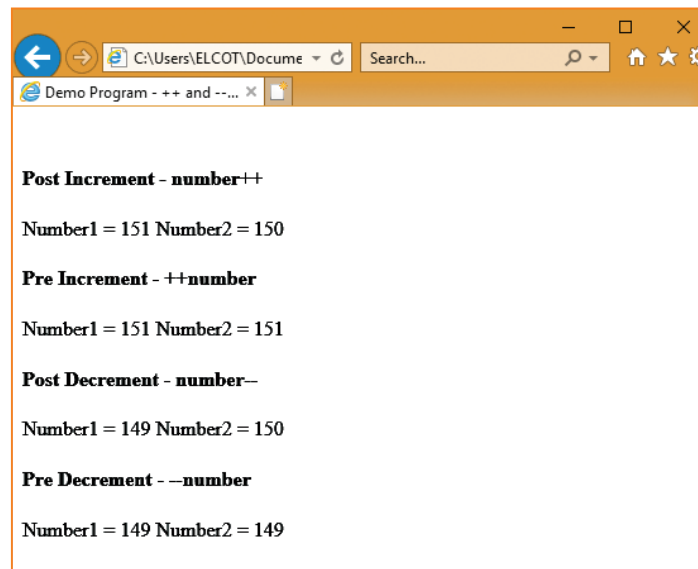
**var m = 2, n = --m; // m and n are both 1**

**var m = 2, n = m--; // n is 2, m is 1**

#### Illustration 14.8 Using ++ and -- Operator – both Prefix and Suffix

```
<Html>
<Head>
    <Title>Demo Program - ++ and -- Operators in JavaScript </Title>
</Head>
<Body>
    <script language="javascript" type="text/javascript">
        var number1 = 150;
        var number2 = number1++;
        document.write("<br><h4>Post Increment - number++</h4>");
        document.write("Number1 = "+number1+" Number2 = "+number2);
        document.write("<br><h4>Pre Increment - ++number</h4>");
        var number1 = 150;
        var number2 = ++number1;
        document.write("Number1 = "+number1+" Number2 = "+number2);
        var number1 = 150;
        var number2 = number1--;
        document.write("<br><h4>Post Decrement - number--</h4>");
        document.write("Number1 = "+number1+" Number2 = "+number2);
        document.write("<br><h4>Pre Decrement - --number</h4>");
        var number1 = 150;
        var number2 = --number1;
        document.write("Number1 = "+number1+" Number2 = "+number2);
    </script>
</Body>
</Html>
```

## Output:



```
Post Increment - number++
Number1 = 151 Number2 = 150
Pre Increment - ++number
Number1 = 151 Number2 = 151
Post Decrement - number--
Number1 = 149 Number2 = 150
Pre Decrement - --number
Number1 = 149 Number2 = 149
```

### 14.6.7 Unary + and - Operator:

+ has no effect on numbers but causes non-numbers to be converted into numbers

– Negation (changes the sign of the number or converts the expression to a number and then changes its sign)

### 14.6.8 typeof Operator:

The **typeof** operator is used to get the data type (returns a string) of its operand. The operand can be either a literal or a data structure such as a variable, a function, or an object. The operator returns the data type.

#### Syntax

**typeof operand**

**or**

**typeof(operand)**

typeof returns: boolean, function, number, string, and undefined. The following table summarizes possible values returned by the typeof operator.

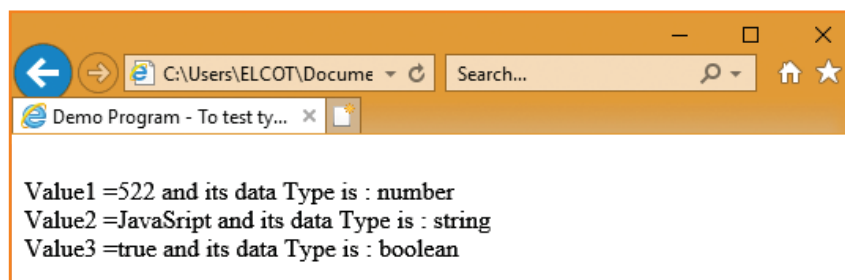
**Table: 14.5**

Type of Operand	Result
Number	“number”
Boolean	“Boolean”
String	“string”
Functions	“function”
Undefined	“undefined”

### Illustration 14.9 typeof operator

```
<Html>
<Head>
  <Title>Demo Program - To test typeof Operator in JavaScript </Title>
</Head>
<Body>
  <script language="javascript" type="text/javascript">
    var value1 = 522, value2="JavaSript"; value3=true;
    document.write("<br>Value1 =" +value1+" and its data Type is : "+typeof(value1));
    document.write ("<br>Value2 =" +value2+" and its data Type is : "+typeof(value2));
    document.write ("<br>Value3 =" +value3+" and its data Type is : "+typeof(value3));
  </script>
</Body>
</Html>
```

### Output:



### 14.6.9 Conditional Operator (?:)

The `?:` is the conditional operator in JavaScript, which requires three operands, hence it is called the ternary operator. The syntax is

**var variablename=(condition) ? value1 : value2;**

In the syntax condition may be relational expression or logical expression. First condition will be evaluated, if the condition returns true then the value of the left side of the colon is assigned to the variable otherwise the value of the right side of the colon will be assigned the variable. For example,

**var result=(10>15) ?100 :150;**

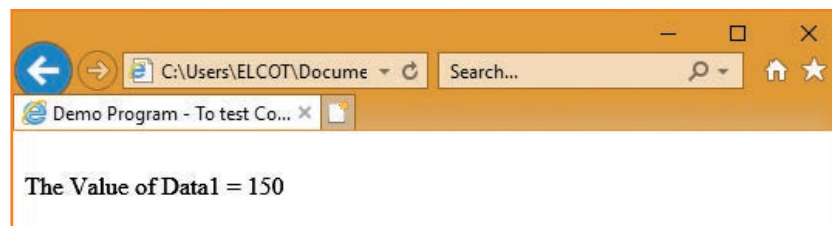
In the above example, since the condition returns false the value 150 will be assigned to result.



### Illustration 14.10 Conditional Operator

```
<Html>
<Head>
  <Title>Demo Program - To test Conditional Operator in JavaScript </Title>
</Head>
<Body>
  <script language="javascript" type="text/javascript">
    var value1 = 522, value2=150, value3;
    value3=(value1<value2) ? value1: value2;
    document.write("<br>The Value of Data1 = "+value3);
  </script>
</Body>
</Html>
```

#### Output:



## 14.7 JavaScript Popup or Dialog Boxes:

JavaScript supports three important types of dialog boxes. Dialog boxes are also called as Popup Boxes. These dialog boxes can be used to raise an alert, or to get confirmation on any input or to have a kind of input from the users. JavaScript supports three kind of popup boxes: Alert box, Confirm box, and Prompt box.

### 14.7.1 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.

#### The syntax of alert box is

Alert("Message");

(or) Window.alert("Message");

Example:

alert("Name is compulsory entry");

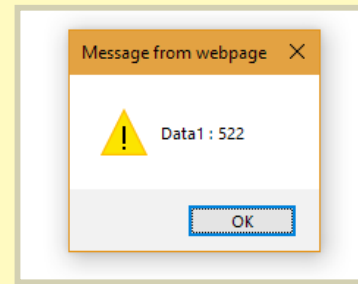
(or) window.alert("Name is compulsory entry");



#### Illustration 14.11 Alert Dialog Box

```
<Html>
<Head>
    <Title>Demo Program - To test Alert Dialog Box in JavaScript </Title>
</Head>
<Body>
    <script language="javascript" type="text/javascript">
        var value1 = 522, value2=10;
        window.alert("Data1 : "+value1);
        alert("Data1 :"+value2);
    </script>
</Body>
</Html>
```

Output:



#### 14.7.2 Confirm 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 `confirm()` will return `true`. If the user clicks on the **Cancel** button, then `confirm()` returns `false`.

**The syntax of confirm dialog box is**

```
confirm("message");
(or)
window.confirm("message");
```

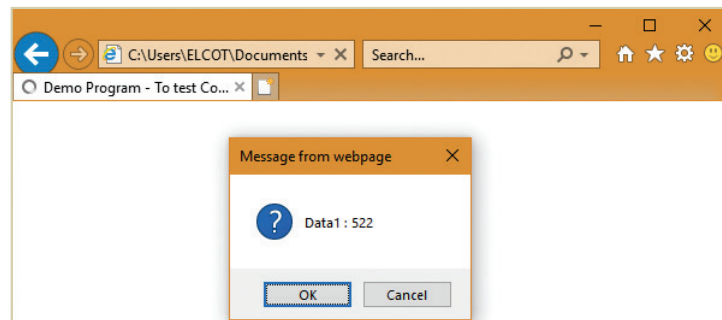
Example:

```
confirm("Hai Do you want to continue:");
(or)
window.confirm("Hai Do you want to continue:");
```

#### Illustration 14.12 Confirm Dialog Box

```
<Html>
<Head>
    <Title>Demo Program - To test Confirm Dialog Box in JavaScript </Title>
</Head>
<Body>
    <script language="javascript" type="text/javascript">
        var value1 = 522, value2=10;
        window.confirm("Data1 : "+value1);
        confirm("Data2 :"+value2);
    </script>
</Body>
</Html>
```

## Output:



### 14.7.3 Prompt Dialog Box:

The prompt dialog box is very useful when the user want to pop-up a text box to get user input. Thus, it enables you to interact with the user. The user needs to fill in the text box field and then click OK.

The prompt dialog box is displayed using a method called `prompt()` which takes two parameters: (i) a label which you want to display in the text box and (ii) a default string to display in the text box. This dialog box has two buttons: OK and Cancel. If the user clicks the OK button, the `prompt()` will return the entered value from the text box. If the user clicks the Cancel button, the `prompt()` returns null. The Syntax of prompt dialog box is,

**Prompt("Message","default Value");**

(or)

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

#### Example:

`prompt("Enter Your Name:","Name");`

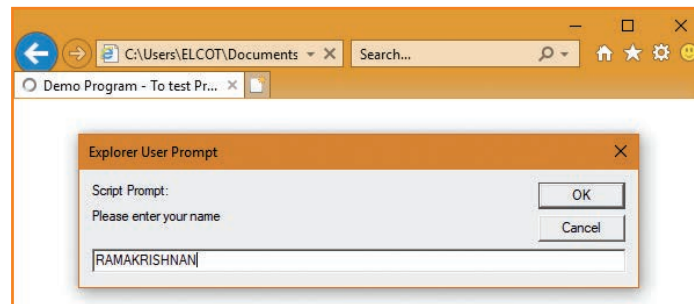
(or)

`window.prompt("Enter Your Name:","Name");`

#### Illustration 14.13 Prompt Dialog Box

```
<Html>
<Head>
  <Title>Demo Program - To test Prompt Dialog Box in JavaScript </Title>
</Head>
<Body>
  <script language="javascript" type="text/javascript">
    var sname = prompt("Please enter your name", "Name");
  </script>
</Body>
</Html>
```

## Output:



## 14.8 Comments in JavaScript:

Very important aspect of good programming style is to insert remarks and commentary directly in source code, making it more readable to yourself as well as to others. Any comments you include will be ignored by the JavaScript interpreter. There are two types of comments, **Single line** and **Multiple lines** comments. Single-line comments begin with a double slash (`//`), causing the interpreter to ignore everything from that point to the end of the line. Multiple line comments begins with `/*` and ends with `*/`.

### For example:

```
// JavaScript single line comment//
```

Multiple line comments begins with `/*` and ends with `*/`

### Points to Remember:

- The `<script>` tag alerts the browser program to start interpreting all the text between these tags as a script
- JavaScript is a case sensitive language
- The scope of a variable is the life time of a variable of source code in which it is defined
- A literal is a fixed value given to a variable in source code.
- An expression is a code fragment that can be evaluated to some data type the language supports.
- Java Script supports all basic arithmetic operators like `+`, `-`, `*`, `/` & `%`
- Java scripts has an in built feature of concatenating strings.
- The types of operator is used to get the data type of a variable.
- JavaScript supports important types of dialog boxes also called as pop up boxes, alert box, confirm dialog box and prompt dialog box.
- `?:` is the conditional operator in Java Script also called as ternary operator.



## Evaluation



### Part-I

Choose the correct answer:

1. Which provided a common scripting language to web developers to design, test and deploy Internet Application  
A) C                                      B) C++                                      C) Java                                      D) JavaScript
2. Expand CGI  
A) Common Gateway Interface                                      B) Complex Gateway Information  
C) Common Gateway Information                                      D) Complex Gateway Interface
3. JavaScript programming language is used to develop the  
A) Dynamic Web Page                                      B) Window                                      C) Web Page                                      D) Home Page
4. The Dynamic Web Page help to save server's  
A) Work                                      B) Route                                      C) Traffic                                      D) Pvath
5. User entered data, is validated before sending it to server is called  
A) Server traffic                                      B) Dynamic Web Page  
C) Server Route                                      D) Web server
6. Java Scripts can be implemented using which statements?  
A) <head>                                      B) <Java>                                      C) <script>                                      D) <text>
7. Expand  
A) Distance Hyper Text Markup language                                      B) Dynamic Hyper Text Markup language  
C) Distance High Text Markup language                                      D) Dynamic High Text Markup language
8. How many attributes specifies that <script> tag in the scripting  
A) 2                                      B) 3                                      C) 4                                      D) 5
9. Which attribute is used to indicate the scripting language and its value should be sent to "Text/JavaScript"  
A) Language                                      B) Text1                                      C) Type                                      D) Body
10. The file reload into the browser use the shortcut key is  
A) F2                                      B) F3                                      C) F4                                      D) F5
11. JavaScript ignores spaces that appear between  
A) Command                                      B) Scripts                                      C) Tokens                                      D) Text
12. A Data value for variable that appears directly in a program by using a  
A) loop                                      B) Literal                                      C) Statement                                      D) Text
13. Which is mostly used to give a warning message to users?  
A) Alert Dialog Box                                      B) Confirm box  
C) Prompt box                                      D) Display box



14. In the below snippet, value of x is    `var x = 250 + 2 - 200;`  
A) 50                                      B) 52                                      C) 48                                      D) 42

### Part – II

#### Very Short Answers

1. Write a syntax of <script> tag
2. What is scope of variables and types of scope variable?
3. Write a notes to type casting in JavaScript
4. How many Literals in JavaScript and mention its types.
5. What is conditional operator give suitable example.
6. What are the comments in Java Script?
7. Write note on types of Operator.
8. Write the role of variable in JavaScript.
9. What is the uses of prompt dialog box?

### Part-III

#### Short Answers

1. What are the advantages of programming language?
2. Brief the basic data types in Java Scripts.
3. Write note on string Operator.
4. Write about <script> tag
5. What are the uses of Logical Operators?
6. Difference between the increment and Decrement operator.

### Part –IV

#### Explain in Detail

1. Explain about the popup dialog boxes in JavaScript.
2. Explain about the Arithmetic operator with suitable example.

#### Case study:

Develop a program for online registration form with some of the client side validation features

#### Reference :

1. Title - Pure JavaScript Author : R.Allen Wyke, Jason D. Gilliam and Charlton Ting,  
Publisher : Techmedia
2. Computer Application Text Book – Govt. of Kerala