Greeting!

UNIT IV – JavaScript

Chapter 14: Introduction to JavaScript

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

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.

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".

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 Crome, etc.,

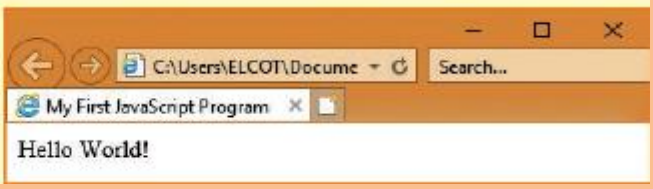
If this is a new document, open the file via browser's **Open Menu.** If the document is already loaded in the Memory, to reload the file into the browser use **"Refresh"** or press **F5** button.

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:



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

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.

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.

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.

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.

Assigning values to variables

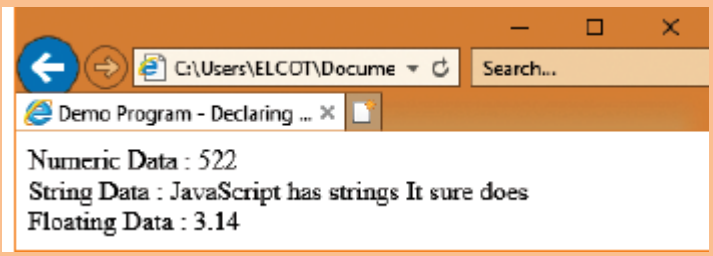
Variables can be assigned initial values when they are declared as follows:

```
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”;

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);

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:

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

Arithmetic Operators

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

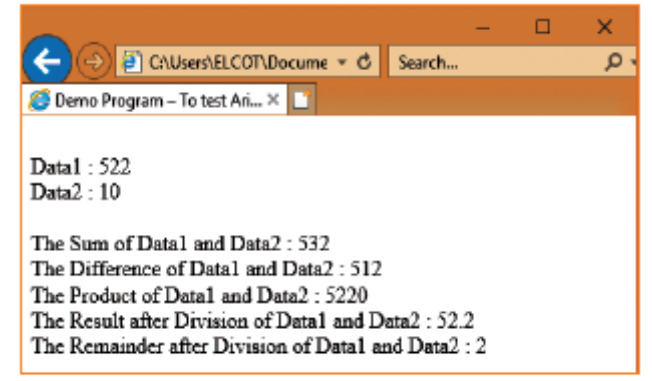
Table: 14.1 – Arithmetic Operators

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



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

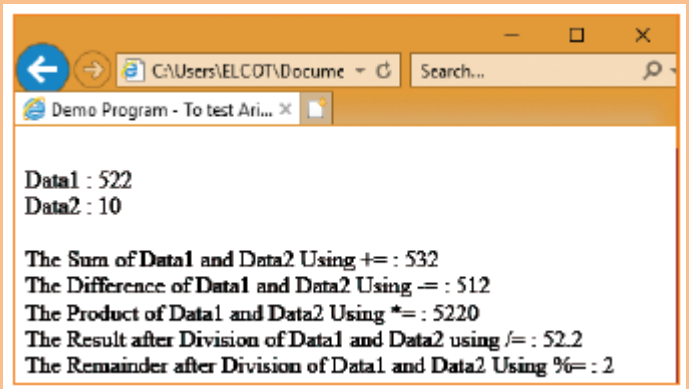
Shorthand Arithmetic Operator	Meaning	Example	Result
+=	Add and assign	var sum = 120; sum += 20;	Variable sum = 140
-=	Subtract and assign	var diff = 120; diff –= 20;	Variable diff = 100
*=	Multiply and assign	var prod = 100; prod *=10;	Variable prod = 1000
/=	Division	Var res = 522; Res/=100	Variable 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:



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: