Unit-4 Client Side Scripting with JavaScript

Structure of JavaScript Program:

JavaScript is a scripting language that can be used to create interactive and dynamic websites. It's a client-side scripting language which runs in the user's web browser. JavaScript files have a .js extension. It is a case sensitive language. JavaScript is not java. To run the .js file in a HTML page we must reference the JavaScript file using a <script> HTML tag. The <script> tags should go inside the page <head> tag. The script tag has an src attribute that points to the JavaScript file.

Example :

```
<html>
<head>
<title>Insert title here</title>
<script type="text/javascript" src="javascriptexample.js"></script>
</head>
<body>
        <h1>Page heading</h1>
</body>
</html>
```

Variables and Data types:

A 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. There are two types of variables in JavaScript local variable and global variable.

Rules for declaring a JavaScript variable:

- Name must start with a letter (a to z or A to Z), underscore (_), or dollar($) sign.
- A variable name cannot start with a number, for example value1.
- Can't user reserved keywords
- JavaScript variables are case sensitive.

There are three ways to declare variable

var: if we declare a variable from var, then we can also declare it again with the same name, and if we want to re-assign its value then we can do that too.

let: if we declare a variable with let, then we can't declare it again with the same name, but can re-assign its value.

const: if we declare variable with const, then we can't change the value

For example of var:

```
<html>
<title>Variables</title>
<head>
</head>
<body>
        <script type="text/javascript">
                var str = "Nawaraj";
                var str = 99;
                document.write(str);
        </script>
</body>
</html>
```

For example of let:

```
<html>
<title>Variables</title>
<head>
</head>
<body>
        <script type="text/javascript">
                let str = "Nawaraj";
                str="Prajapati"
                let str = 99;       //not possible
                document.write(str);
        </script>
</body>
</html>
```

For example of const:

```
<html>
<title>Variables</title>
<head>
</head>
<body>
        <script type="text/javascript">
                const str = "Nawaraj";
                const str="Prajapati";    //not possible
                str = "Hello";              // not possible
                document.write(str);
        </script>
</body>
</html>
```

There are two type of variables:

1. Local variable
2. Global variable

Local variable:

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

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

Global variable

Global variable can be accessed from any function. For example:

```
<script>
        var value = 50;                 //global variable
        function a() {
                alert(value);
        }
        function b() {
                document.write("Welcome to javascript");
        }
</script>
```

Data Types:

JavaScript provides different **data types** to hold different types of values. There are two types of data types in JavaScript.

1. Primitive data type
2. Non-primitive (reference) data type

Primitive data types:

There are five types of primitive data types in JavaScript.

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

JavaScript non-primitive data types:

Data Type	Description
Object	represents instance through which we can access members
Array	represents group of similar values

Example of primitive data types:

```
<html>
<title>Variables</title>
<head>
</head>
<body>
        <script type="text/javascript">
                var str = "Nawaraj";
                str=99;
                document.write(str);
                document.write("<br>");
                document.write(typeof str);
        </script>
</body>
</html>
```

Output:

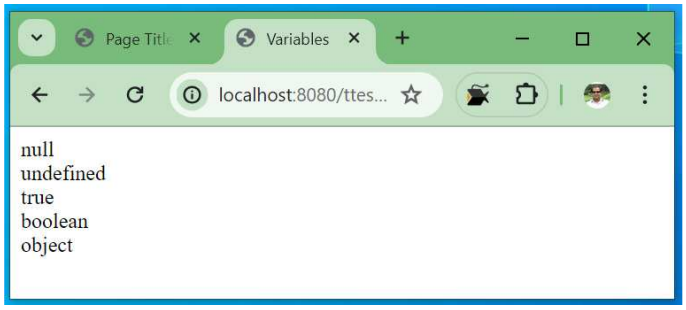
99
number

Another example:

```
<html>
<title>Variables</title>
<head>
</head>
<body>
        <script type="text/javascript">
                var a=null;
                var b=undefined;
                var c=true;
                document.write(a);
                document.write("<br>");
                document.write(b);
                document.write("<br>");
                document.write(c);
                document.write("<br>");
                document.write(typeof c);
                document.write("<br>");
                document.write(typeof a);

        </script>
</body>
</html>
```

Output:



Example of non-primitive data types:

```
<html>
<title>Variables</title>
<head>
</head>
<body>
        <script type="text/javascript">
                var a = ["Nawaraj","Abhiraj","Prajapati"];
                document.write(a);
                document.write("<br>");
                document.write(typeof a);

        </script>
</body>
</html>
```

Output:

Nawaraj,Abhiraj,Prajapati
object

```
<html>
<title>Variables</title>
<head>
</head>
<body>
        <script type="text/javascript">
                var name = {
                                firstname:"Nawaraj",
                                lastname:"Prajapati"
                };
                document.write(name);
                document.write("<br>");
                document.write(typeof name);

        </script>
</body>
</html>
```

Output:

[object Object]
object

Statements:

A JavaScript program is a sequence of statements. Each statement is an instruction for the computer to do something. A JavaScript program is a list of programming statements, JavaScript statements are compose of values, operators, expressions, keywords and comments.

Expressions:

An expression is any valid set of literals, variables, operators, and expressions that evaluates to a single value; the value can be a number, a string, or a logical value. Conceptually, there are two types of expressions, those that assign a value to a variable, and those that simply have a value.

For example: The expression x = 7 is an expression that assigns x the value seven. This expression itself evaluates to seven. Such expressions use assignment operators. On the other hand, the expression 3 + 4 simply evaluates to seven; it does not perform an assignment. The operators used in such expressions are referred to simply as operators.

Keywords:

Keywords are the reserved word in JavaScript that we can't use to indicate variable and function names. There are 63 keywords that JavaScript provides.

abstract	arguments	boolean	break
byte	case	catch	char
const	continue	debugger	default
delete	do	double	else
eval	false	final	finally
float	for	function	goto
if	implements	in	instanceof
int	interface	let	long
native	new	null	package
private	protected	public	return
short	static	switch	synchronized
this	throw	throws	transient
true	try	typeof	var
void	volatile	while	with
yield			

Operators:

An operator is a symbol that is used to perform operations according to user requirements in programs. We can perform this in variables or values.

Operators can be classified into the following types:

- Arithmetic Operators
- Assignment Operators
- Relational Operators
- Logical Operators
- Conditional Operator

Arithmetic Operators (+, -, *, /, %):

The arithmetic operators are the symbols that are used to perform basic mathematical operations like addition, subtraction, multiplication, division and percentage modulo.

Operator	Meaning	Example
+	Addition	10 + 5 = 15
-	Subtraction	10 - 5 = 5
*	Multiplication	10 * 5 = 50
/	Division	10 / 5 = 2
%	Remainder of the Division	5 % 2 = 1

For example:

Assignment Operators (=, +=, -=, *=, /=, %=):

The assignment operators are used to assign right-hand side value (Rvalue) to the left-hand side variable (Lvalue). The assignment operator is used in different variants along with arithmetic operators.

Operator	Meaning	Example
=	Assign the right-hand side value to left-hand side variable	A = 15
+=	Add both left and right-hand side values and store the result into left-hand side variable	A += 10 ⇒ A = A+10
-=	Subtract right-hand side value from left-hand side variable value and store the result into left-hand side variable	A -= B ⇒ A = A-B
*=	Multiply right-hand side value with left-hand side variable value and store the result into left-hand side variable	A *= B ⇒ A = A*B
/=	Divide left-hand side variable value with right-hand side variable value and store the result into the left-hand side variable	A /= B ⇒ A = A/B
%=	Divide left-hand side variable value with right-hand side variable value and store the remainder into the left-hand side variable	A %= B ⇒ A = A%B

For example:

Relational Operators (<, >, <=, >=, ==, !=):

The relational operators are the symbols that are used to compare two values. That means the relational operators are used to check the relationship between two values. Every relational operator has two results TRUE or FALSE. In simple words, the relational operators are used to define conditions in a program.

Operator	Meaning	Example
<	Returns TRUE if the first value is smaller than second value otherwise returns FALSE	10 < 5 is FALSE
>	Returns TRUE if the first value is larger than second value otherwise returns FALSE	10 > 5 is TRUE
<=	Returns TRUE if the first value is smaller than or equal to second value otherwise returns FALSE	10 <= 5 is FALSE
>=	Returns TRUE if the first value is larger than or equal to second value otherwise returns FALSE	10 >= 5 is TRUE
==	Returns TRUE if both values are equal otherwise returns FALSE	10 == 5 is FALSE
!=	Returns TRUE if both values are not equal otherwise returns FALSE	10 != 5 is TRUE

For example:

Logical Operators (&&, ||, !):

The logical operators are the symbols that are used to combine multiple conditions into one condition. The following table provides information about logical operators.

Operator	Meaning	Example
&&	**Logical AND** - Returns TRUE if all conditions are TRUE otherwise returns FALSE	10 < 5 && 12 > 10 is FALSE
||	**Logical OR** - Returns FALSE if all conditions are FALSE otherwise returns TRUE	10 < 5 || 12 > 10 is TRUE
!	**Logical NOT** - Returns TRUE if condition is FLASE and returns FALSE if it is TRUE	!(10 < 5 && 12 > 10) is TRUE

For example:

Conditional operators (Ternary Operator):

The conditional operator is also called a ternary operator because it requires three operands. This operator is used for decision making. In this operator, first, we verify a condition, then we perform one operation out of the two operations based on the condition result. If the condition is TRUE the first option is performed, if the condition is FALSE the second option is performed.

The conditional operator behaves like an if-else statement.

Conditional Operator Syntax:

booleanExpression? expression1: expression2

For example:

```
<html>
<head>
<title>Page Title</title>
</head>
<body>
        <script>
                var a = 50, b = 10, c = 20, large;

                large = (a > b) ? (a > c ? a : c) : (b > c ? b : c);

                document.write("<br>The Largest Number is = " + large);
        </script>
</body>
</html>
```

Output: