JAVASCRIPT

Introduction

JavaScript is a lightweight, interpreted programming language with object-oriented capabilities that allows you to build interactivity into otherwise static HTML pages.

> *JavaScript code is not compiled but translated by the translator. This translator is embedded into the browser and is responsible for translating javascript code.*

Key Points

- It is Lightweight, interpreted programming language.
- It is designed for creating network-centric applications.
- It is complementary to and integrated with Java.
- It is complementary to and integrated with HTML
- It is an open and cross-platform

JavaScript Statements

JavaScript statements are the commands to tell the browser to what action to perform. Statements are separated by semicolon ; .

> *JavaScript statement constitutes the JavaScript code which is translated by the browser line by line.*

Example of JavaScript statement:

```
document.getElementById("demo").innerHTML = "Welcome";
```

Following table shows the various JavaScript Statements −

Sr.No.	Statement	Description
1.	switch case	A block of statements in which execution of code depends upon different cases. The interpreter checks each case against the value of the expression until a match is found. If nothing matches, a **default** condition will be used.
2.	If else	The **if** statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.
3.	While	The purpose of a while loop is to execute a statement or code block repeatedly as long as expression is true. Once expression becomes false, the loop will be exited.
4.	do while	Block of statements that are executed at least once and continues to be executed while condition is true.
5.	for	Same as while but initialization, condition and increment/decrement is done in the same line.

6.	for in	This loop is used to loop through an object's properties.
7.	continue	The continue statement tells the interpreter to immediately start the next iteration of the loop and skip remaining code block.
8.	break	The break statement is used to exit a loop early, breaking out of the enclosing curly braces.
9.	function	A function is a group of reusable code which can be called anywhere in your programme. The keyword function is used to declare a function.
10.	return	Return statement is used to return a value from a function.
11.	var	Used to declare a variable.
12.	try	A block of statements on which error handling is implemented.
13.	catch	A block of statements that are executed when an error occur.
14.	throw	Used to throw an error.

JavaScript Comments

JavaScript supports both C-style and C++-style comments, thus:

- 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 //-->.

Example

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

// this is a comment. It is similar to comments in C++

/*
 * This is a multiline comment in JavaScript
 * It is very similar to comments in C Programming
 */
//-->
<script>
```

JavaScript variable

Variables are referred as named containers for storing information. We can place data into these containers and then refer to the data simply by naming the container.

Rules to declare variable in JavaScript

Here are the important rules that must be followed while declaring a variable in JavaScript.

- In JavaScript variable names are case sensitive i.e. a is different from A.
- Variable name can only be started with a underscore _ or a letter *fromatozorAtoZ*, or dollar $ sign.

- Numbers $0 to 9$ can only be used after a letter.
- No other special character is allowed in variable name.

Before you use a variable in a JavaScript program, you must declare it. Variables are declared with the var keyword as follows −

```
<script type="text/javascript">
<!--
var money;
var name, age;
//-->
</script>
```

Variables can be initialized at time of declaration or after declaration as follows −

```
<script type="text/javascript">
<!--
var name = "Ali";
var money;
money = 2000.50;
//-->
</script>
```

Javascript Data Type

There are two kinds of data types as mentioned below −

- Primitive Data Type
- Non Primitive Data Type

The following table describes **Primitive Data Types** available in javaScript

Sr.No.	Datatype Description
1.	**String** Can contain groups of character as single value. It is represented in double quotes.E.g. var x= “tutorial”.
2.	**Numbers** Contains the numbers with or without decimal. E.g. var x=44, y=44.56;
3.	**Booleans** Contain only two values either true or false. E.g. var x=true, y= false.
4.	**Undefined** Variable with no value is called Undefined. E.g. var x;
5.	**Null** If we assign null to a variable, it becomes empty. E.g. var x=null;

The following table describes **Non-Primitive Data Types** in javaScript

Sr.No.	Datatype Description
1.	**Array** Can contain groups of values of same type. E.g. var x={1,2,3,55};
2.	**Objects** Objects are stored in property and value pair. E.g. var rectangle = { length: 5, breadth: 3};

JavaScript Functions

Function is a group of reusable statements *Code* that can be called any where in a program. In javascript function keyword is used to declare or define a function.

Key Points

- To define a function use function keyword followed by functionname, followed by parentheses .
- In parenthesis, we define parameters or attributes.
- The group of reusabe statements *code* is enclosed in curly braces {}. This code is executed whenever function is called.

Syntax

```
function functionname (p1, p2) {
 function coding…
}
```

JavaScript Operators

Operators are used to perform operation on one, two or more operands. Operator is represented by a symbol such as +, =, *, % etc. Following are the operators supported by javascript −

- Arithmetic Operators
- Comparison Operators
- Logical *orRelational* Operators
- Assignment Operators
- Conditional *orternary* Operators
- Arithmetic Operators

Arithmatic Operators

Following table shows all the arithmetic operators supported by javascript −

Operator	Description	Example
+	Add two operands.	10 + 10 will give 20
-	Subtract second operand from the first.	10 – 10 will give 0
*	Multiply two operands.	10 * 30 will give 300
/	Divide numerator by denominator	10/10 will give 1
%	It is called modulus operator and gives remainder of the	10 % 10 will give 0

	division.	
++	Increment operator, increases integer value by one	10 ++ will give 11
--	Decrement operator, decreases integer value by one	10 – will give 9

Comparison Operators

Following table shows all the comparison operators supported by javascript −

Operator	Description	Example
==	Checks if values of two operands are equal or not, If yes then condition becomes true.	10 == 10 will give true
!=	**Not Equal to operator** Checks if the value of two operands is equal or not, if values are not equal then condition becomes true.	10 !=10 will give false
>	**Greater Than operator** Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	20 > 10 will give true
<	**Less than operator** Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	10 < 20 will give true
>=	**Greater than or equal to operator** Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	10 >=20 will give false
<=	**Less than or equal to operator** Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	10 <=20 will give true.

Logical Operators

Following table shows all the logical operators supported by javascript −

Operator	Description	Example
&&	Logical **AND** operator returns true if both operands are non zero.	10 && 10 will give true.
||	Logical **OR** operator returns true If any of the operand is non zero	10 || 0 will give true.
!	Logical **NOT** operator complements the logical state of its operand.	! 10 && 10 will give false.

Assignment Operators

Following table shows all the assignment operators supported by javascript −

Operator	Description	Example
=	**Simple Assignment operator**	C = A + B will assign

	Assigns values from right side operands to left side operand.	value of A + B into C
+=	**Add AND assignment operator** It adds right operand to the left operand and assign the result to left operand	C += A is equivalent to C = C + A
-=	**Subtract AND assignment operator** It subtracts right operand from the left operand and assign the result to left operand	C -= A is equivalent to C = C - A
*=	**Multiply AND assignment operator** It multiplies right operand with the left operand and assign the result to left operand	C *= A is equivalent to C = C * A
/=	**Divide AND assignment operator** It divides left operand with the right operand and assign the result to left operand	C /= A is equivalent to C = C / A
%=	**Modulus AND assignment operator** Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A

Conditional Operator

It is also called ternary operator, since it has three operands.

Operator	Description	Example
?:	Conditional Expression	If Condition is true? Then value X : Otherwise value Y

Control Structure

Control structure actually controls the flow of execution of a program. Following are the several control structure supported by javascript.

- if ... else
- switch case
- do while loop
- while loop
- for loop

If ... else

The if statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.

Syntax

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

Example

```
<script type="text/javascript">
<!--
var age = 20;
```

```
if( age > 18 ){
   document.write("<b>Qualifies for driving</b>");
}
//-->
</script>
```

Switch case

The basic syntax of the switch statement is to give an expression to evaluate and several different statements to execute based on the value of the expression. The interpreter checks each case against the value of the expression until a match is found. If nothing matches, a default condition will be used.

Syntax

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

Example

```
<script type="text/javascript">
<!--
var grade='A';
document.write("Entering switch block<br/>");
switch (grade)
{
  case 'A': document.write("Good job<br/>");
            break;
  case 'B': document.write("Pretty good<br/>");
            break;
  case 'C': document.write("Passed<br/>");
            break;
  case 'D': document.write("Not so good<br/>");
            break;
  case 'F': document.write("Failed<br/>");
            break;
  default:  document.write("Unknown grade<br/>")
}
document.write("Exiting switch block");
//-->
</script>
```

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);
```

Example

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

This will produce following result −

```
Starting Loop
Current Count : 0
Loop stopped!
```

While Loop

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

Syntax

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

Example

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

This will produce following result −

```
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!
```

For Loop

The for loop is the most compact form of looping and includes the following three important parts −

- The loop initialization where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.

- The test statement which will test if the given condition is true or not. If condition is true then code given inside the loop will be executed otherwise loop will come out.
- The iteration statement where you can increase or decrease your counter.

Syntax

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

Example

```
<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>
```

This will produce following result which is similar to while loop −

```
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!
```

Creating Sample Program

Following is the sample program that shows time, when we click in button.

```
<html>
<body>
<button onclick="this.innerHTML=Date()">The time is?</button>
<p>Click to display the date.</p>
<button onclick="displayDate()">The time is?</button>
<script>
function displayDate() {
    document.getElementById("demo").innerHTML = Date();
}</script>

<p ></p>

</body>
</html>
```

Output