JavaScript

JavaScript is a lightweight, interpreted programming

Language. It can be used for both Client-side as well as Server-side developments. JavaScript also known as a scripting language for web pages.

JavaScript - Syntax

JavaScript is a case-sensitive language.JavaScript can be implemented using JavaScript statements that are placed within the

<script>... </script> HTML tags in a web page.

You can place the <script> tags, containing your JavaScript, anywhere within your web page

The <script> tag alerts the browser program to start interpreting all the text between these tags as a script. A simple syntax of your JavaScript will appear as follows.

```
<script ...>
  JavaScript code
</script>


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

JavaScript Display Possibilities

JavaScript can "display" data in different ways:

- Writing into an HTML element, using innerHTML.
- Writing into the HTML output using document.write().
- Writing into an alert box, using window.alert().
- Writing into the browser console, using console.log().

```
<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = 5 + 6;

document.write(5 + 6);



window.alert(5 + 6);
</script>

</body>
</html>
```

JavaScript Variables

4 Ways to Declare a JavaScript Variable:

- Using var
- Using let
- Using const
- Using nothing

Types of JavaScript Operators

There are different types of JavaScript operators:

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- String Operators
- Logical Operators
- Bitwise Operators
- Ternary Operators
- Type Operators

JavaScript Data Types

JavaScript variables can hold many data types: numbers, strings, objects and more:

```
var length = 16;                          // Number
var firstName = "Sachin";                 // String
var x = {firstName:"Sachin", lastName:"Raj"};    // Object
```

<u>JavaScript has 8 Datatypes</u>

1. String
2. Number
3. Bigint
4. Boolean
5. Undefined
6. Null
7. Symbol
8. Object

- <u>Seven primitive data types:</u>
  - number for numbers of any kind: integer or floating-point
  - bigint for integer numbers of arbitrary length.
  - string for strings. A string may have zero or more characters, there's no separate single-character
  - type.boolean for true/false.
  - null for unknown values – a standalone type that has a single value null.
  - undefined for unassigned values – a standalone type that has a single value undefined.
  - symbol for unique identifiers

    for example

```
// creating symbol
const x = Symbol()
document.write(typeof x)
```

// symbol

- And one non-primitive data type:
  - object for more complex data structures.

JavaScript Types are Dynamic

JavaScript has dynamic types. This means that the same variable can be used to hold different data types:

Example

```
let x;       // Now x is undefined
x = 5;       // Now x is a Number
x = "sachin";  // Now x is a String
```

JavaScript Objects

JavaScript objects are written with curly braces.

Object properties are written as name:value pairs, separated by comma

------

Example:

```
<!DOCTYPE html>

<html>

<body>

<p id="demo"></p>
```

```
<script>
var person = {
   firstName : "Akhil",
   lastName  : "kumar",
   age       : 20,
   eyeColor  : "blue"
};
document.getElementById("demo").innerHTML =
person.firstName + " is " + person.age + " years old.";
</script>

</body>
</html>
```

JavaScript Arrays

JavaScript arrays are written with square brackets.

Array items are separated by commas.

Example:

```
<html>
<body>
<p id="demo"></p>
```

```
<script>
var cars = ["Honda","Volvo","BMW"];
document.getElementById("demo").innerHTML =
cars[0];
</script>

</body>
</html>
```

<u>Using the JavaScript Keyword new</u>

The following example also creates an Array, and assigns values to it:

Example

```
var cars = new Array("Honda", "Volvo", "BMW");
```

<u>Example</u>

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Arrays</h2>
<p id="demo"></p>
<script>
var cars = new Array("Honda ", "Volvo", "BMW");
```

```
document.getElementById("demo").innerHTML = cars;
</script>
</body>
</html>
```

The if Statement

Syntax

```
if (condition)
 {
   block of code to be executed if the condition is true
}
else
 {
   block of code to be executed if the condition is false
}
```

The else if Statement

Use the else if statement to specify a new condition if the first condition is false.

Syntax

```
if (condition1) {
   block of code to be executed if condition1 is true
} else if (condition2) {
   block of code to be executed if the condition1 is false and condition2 is true
} else {
   block of code to be executed if the condition1 is false and condition2 is false
}
```

Example:

```
let day=new Date().getDay();

if(day==0)
{
document.write("Today is Sunday");
 }
else if(day==1)
{
document.write("Today is Monday");
 }
else if(day==6)
{
document.write("Today is Saturday");
 }
```

```
else
{
document.write("Looking forward to the Weekend");
 }
```

JavaScript Switch Statement

Use the switch statement to select one of many blocks of code to be executed.

Syntax

```
switch(expression)
{
   case n:
      code block
      break;
   case n:
      code block
      break;
   default:
      code block
}
```

<u>Example</u>

```
<html>
<body>
<script>
switch (new Date().getDay())
{
   case 0:
      text = "Today is Sunday";
      break;
case 2:
      text = "Today is Tuesday";
      break;
case 6:
      text = "Today is Saturday";
      break;
   default:
      text = "Looking forward to the Weekend";
}
document.write(text);
</script>
</body>
```

```
</html>
```

For Loop

he syntax of for loop is JavaScript is as follows -

```
for (initialization; test condition; iteration statement)
{

  Statement(s) to be executed if test condition is true
}
```

Example

```
<html>
  <body>

    <script type="text/javascript">

        var count;
        document.write("Starting Loop" + "<br />");

        for(count = 0; count < 10; count++)
      {
          document.write("Current Count : " + count );
```

```
            document.write("\n");
          }

          document.write("Loop stopped!");
        </script>
      </body>
</html>
```

while Loop

The syntax of while loop in JavaScript is as follows -

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

Example

```
<html>
  <body>

    <script type="text/javascript">

          var count = 0;
          document.write("Starting Loop ");
          while (count < 10)
   {
 document.write("Current Count : " + count + "<br />");
           count++;
          }

          document.write("Loop stopped!");

     </script>

   </body>
</html>
```

do...while Loop

The syntax for do-while loop in JavaScript is as follows -

```
do{
  Statement(s) to be executed;
} while (expression);
```

Example

```
<html>
   <body>
```

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

          var count = 0;

          document.write("Starting Loop" + "<br />");
          do
         {
document.write("Current Count : " + count + "<br />");
            count++;
          }  while (count < 5);
          document.write ("Loop stopped!");

      </script>

      <p>Set the variable to different value and then
try...</p>
   </body>
```

Loop Control

JavaScript provides full control to handle loops and switch statements. There may be a situation when you need to come out of a loop without reaching its bottom. There may also be a situation when you want to skip a part of your code block and start the next iteration of the loop.

To handle all such situations, JavaScript provides break and continue statements. These statements are used to immediately come out of any loop or to start the next iteration of any loop respectively.

The break Statement

Example

```
var x = 1;
 document.write("Entering the loop<br /> ");


 while (x < 20) {
   if (x == 5) {
     break;   // breaks out of loop completely
   }
   document.write( x + "<br />");
   x = x + 1;
 }
 document.write("Exiting the loop!<br /> ");
```

//----------------------------

Output

```
Entering the loop
1
```

```
2
3
4
Exiting the loop!
```

<u>The continue Statement</u>

```
 var x = 0;
 document.write("Entering the loop<br /> ");
  while (x < 10) {
  x = x + 1;
        if (x == 5) {
         continue;   // skip rest of the loop body
         }
  document.write( x + "<br />");
  }
  document.write("Exiting the loop!<br /> ");
//------------------------
```

Output

```
Entering the loop
1
2
3
4
```

```
6
7
8
9
10
Exiting the loop!
```

Functions

A function is a block of code that performs a specific task. A function is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing the same code again and again. It helps programmers in writing modular codes.ie Functions allow a programmer to divide a big program into a number of small and manageable functions.

Advantages of functions.

1. Code reusability: We can call a function several times so it save coding.
2. Less coding: It makes our program compact. We don't need to write many lines of code each time to perform a common task.

Function Syntax

```
function functionname(parameter-list)
 {
   statements
 }
```

Example

```
<html>
  <head>
        <script type="text/javascript">
        function say()
        {
          document.write ("Hello there!");
        }
      </script>
        </head>
  <body>
 <script type="text/javascript">
     say();
 </script>
  </body>
</html>
```

Function Arguments

We can call function by passing arguments

Example

```
<html>
  <head>
 <script>
```

```
        function sum(a,b)
        {
         r=a+b;
          document.write ("sum="+r);
        }
      </script>


   </head>
   <body>
    <script>
x=eval(prompt("enter a number"));
y=eval(prompt("enter another number"));
  sum(x,y)
 </script>
   </body>
</html>
```

<u>Function Return</u>

The return statement can be used to return the value to a function call.

The return statement denotes that the function has ended. Any code after return is not executed.

Example

```
<html>
   <head>
 <script>
         function sum(a,b)
         {
          r=a+b;
          return r;
         }
      </script>
</head>
   <body>
    <script>
x=eval(prompt("enter a number"));
y=eval(prompt("enter another number"));
  result=sum(x,y)
 document.write ("sum="+  result);
 </script>
   </body>
</html>
```