Variable Declaration

var:

This statement is used to declare a variable. If you use it within a function, the variable is guaranteed to be local to that function. If you use it outside the function, the variable is considered global.

Example:

```
var count = 30, length = 1;
```

Because **JavaScript is a loosely typed language**, you do not need to specify the type when you declare the variable.

A variable is also automatically declared the first time you assign it a value.

Using **var** will help avoid conflicts between local and global variables.

If you re-declare a JavaScript variable, it will not lose its value.

JavaScript Data Types (Subtypes)

In JavaScript there are 5 different data types that can contain values:

- string
- number
- boolean
- object
- function

There are 3 types of objects:

- Object
- Date
- Array

And 2 data types that cannot contain values:

- null
- undefined

Using **typeof(var)** we can find the subtype (string, number, boolean, object) of the variable.

```
typeof "John"                  // Returns string
typeof 3.14                    // Returns number
typeof false                   // Returns boolean
typeof [1, 2, 3, 4]            // Returns object
typeof { name: 'John', age: 34 }   // Returns object
```

A variable without the value is *undefined.*

"use strict";

It is not a statement, but a literal expression, ignored by earlier versions of JavaScript (prior to 1.8.5)

Supported in IE10+, FF4+, Chrome 13+...

- The purpose of "use strict" is to indicate that the code should be executed in "strict mode".
- It should be added to the beginning of a JavaScript file (global effect), or a JavaScript function (local effect).
- Using a variable (property or object) without declaring it, is not allowed.
- with statement is not allowed.
- The string "eval" and "arguments" cannot be used as a variable.

```
x = 3.14;      // This will not cause an error.
myFunction();  // This will cause an error


function myFunction() {
  "use strict";
   x = 3.14;
}
```

Operators

Arithmetic Operator: +, -, *, /, %

Assignment operator: =, +=, -=, *=, /=, %=

Logical Operators: &&, ||, !,

Comparison Operator: ==, !=, >, <, >=, <=, ===(equal value and equal type)

Conditional Operator: ?:

Operator precedence: (from lowest precedence to highest)

Assignment operators (=, +=, -=, *=, /=, %=)

Conditional (?:)

Logical or (||)

Logical and (&&)

Equality (==, !=)

Relational (<, <=, >, >=)

Addition/subtraction (+, -)

Multiply/divide/modulus (*, /, %)

Parentheses (())

Control Statements

If..else:

```
if (condition)
  Command;
else
  Command;
```

```
if (condition)
{
  Several lines of JavaScript code
}
else
{
   Several lines of JavaScript code
}
```

Switch:

```
switch(expression)
{
          case var1:
                    statements
                    break;
          case var2:
                    statements
                    break;
          default:
                    statements
}
```

while:

```
while ( cond stmt )
{
  zero of more statements
}
```

for…loop:

```
for ( initstmt; condstmt; updstmt )
{
  statements
}
```

break; continue;

for..in

for (varname **in** arr1)

```
{
  statements
}
```

With:

```
with ( objname )
{
  statements
}
```

Error Handling

```
try {
   Block of code to try
}
catch(err) {
   Block of code to handle errors
}
```

The **try** statement lets you test a block of code for errors.

The **catch** statement lets you handle the error.

The **throw** statement lets you create custom errors.

The **finally** statement lets you execute code, after try and catch, regardless of the result.

Eg:

```
   try {
      if(num == "") throw "its Empty";
      if(isNaN(num)) throw "its not a number";
      if(num > 10) throw "its big";
      if(num < 5) throw "its small";
   }
   catch(err) {
      message.innerHTML = "Input " + err;
   }
   finally {
      document.getElementById("demo").value = "";
   }
```

Understanding Arrays

```
var sampleArray = new Array(size);
sampleArray[0] == "1st Element"
. . .
sampleArray[N-1] == "(N-1)th Element"
```

Array elements are referred to by their indexes i.e. the numbers in brackets. In JavaScript, arrays start with index 0, so the Nth element in an array is actually sampleArray[N-1].

```
var ar = [1, 2, 3]; //To declare and initialize an array.

var sampleArray = "1,2,3".split(",");
alert(sampleArray[0])
alert(sampleArray.join("+"))
alert(sampleArray.length);
sampleArray[15] = "Demo";
alert(sampleArray.length);
```

Note: Multidimensional Arrays are not supported in JavaScript.

Properties	**Description**
length	Gives the length of the Array.

Methods	**Return Type**	**Parameters**
sort	New Array	
reverse	New Array	
join	New Array	Separator String
concat	New Array	Arr1, Arr2, Arr3
valueOf / toString	String	comma separated string
push		Value
pop	Value	
shift	the string that was "shifted out".	
unshift	new array length	
slice		(startIndex, endIndex+1)

myArray.constructor.toString().indexOf("Array") > -1; //Returns true if myArray is an Array.

Function Declaration

```
function  function_name(parameter1, parameter2, parameter3)
{
        command block
}
```

Use return keyword to return a value from the function.

Example:

```
function Add(a, b) {
        return a + b;
}
```

```
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta charset="utf-8" />
  <title></title>
  <script>
    function Sayhello()
    {
      //var name = window.document.forms["form1"].elements["txtName"].value
      var name = document.form1.txtName.value
      alert("Hello " + name)
    }
  </script>
</head>
<body>
  <form name="form1" action="/" method="post">
    <input type="text" name="txtName" value="" />
    <input type="button"  name="btnSayHello" value="Say Hello" onclick="Sayhello()" />
  </form>
</body>
</html>
```