# JAVA SCRIPT NOTES

Basic points:

* JS is "Case-sensitive language"
* JS should be placed inside SCRIPT TAG always
* Script has type attribute and language attribute (but outdated)

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

* We can maintain the script as our wish. (Line by line, with gaps, etc.,)

\*\* But maintaining semicolons after each statement is good practice

* Commenting in JS can be done in 3 ways :
* One line comment : // or <!-- JS //-->
* Multiple line comment :

/\*

JS

\*/

Placement of JS in an html doc

IN <head>:

Useful for giving response when user clicks something. Shortly we can say as "EVENT BASED"

IN <body>:

Useful when we wanna JS to load when we open the page itself. Shortly we can say as "JS will appear in the page when opened"

IN BOTH <head> and <body>: Combination of above two

In an External File:

Useful when we wanna use the same JS repeatedly (REUSABILITY IS ACHIEVED HERE). We have to link JS like this in html doc.

\* External file name must be declared with “.js” extension.

Place this inside <head> Element

<script src="<path>/<filename>.js" />

\*\*\* When using external JS file, script tag should only contain ‘src’ attribute.

JS Data types overview

JS supports the following data types:

1. Primitive Data types (Strings, Numbers, Boolean),
2. Trivial Data types (null, undefined)
3. Composite Data type (Object)

* In primitive data types, Numbers will support "integer and float data types by default"
* In Trivial Data types, null & undefined are there… Each of which defines only one "Single Value"

JS Variables:

* JS Variable must start with either ALPHABET or UNDERSCORE.

Ex: foo, \_dude

* In JS, a variable can also be called as "Named Container".
* Here no need to declare the type of variable as we do in Java

Declaration: A Variable will be declared in JS through 'var' keyword

Ex: var use\_Me;

Initialization: A variable can be initialized at declaration time or later after declaration

Ex:

var hey;

hey="tell me";

var hello="speak";

Scope of the variables (Global and Local):

Global:

Variable declared outside all the functions and inside 'script' element will be considered as global variable.

Local:

Variable declared inside the function will be considered as Local variable.

Ex:

<script type="text/javascript">

//Global variable hi

var hi=123.32;

function touchMe()

{

//Local Variable hello

var hello="greet";

}

</script>

* Each variable must be unique. No variable should be declared twice.

Limitation:

JS Reserved words (Keywords in C) must not be used as Variable names.

|  |  |  |  |
| --- | --- | --- | --- |
| abstract  boolean  break  byte  case  catch  char  class  const  continue  debugger  default  delete  do  double | else  enum  export  extends  false  final  finally  float  for  function  goto  if  implements  import  in | Instanceof  int  interface  long  native  new  null  package  private  protected  public  return  short  static  super | switch  synchronized  this  throw  throws  transient  true  try  typeof  var  void  volatile  while  with |

Operators

* Ex: 3 + 4: Here 3 & 4 are Operands. “+ is an Operator here”

JS supports 5 types of Operators:

1. Arithmetic Operators
2. Comparison Operators
3. Logical (or Relational) Operators
4. Assignment Operators
5. Conditional (or ternary) Operators

Arithmetic Operators:

JS supports the following Arithmetic Operators…

* + (Addition)
* - (Subtraction)
* \* (Multiplication)
* / (Division)
* % (Modulus)
* ++(increment)
* -- (Decrement)
* The difference between ‘Division (Gives quotient) and Modulus (Gives remainder) Operators’ is the following:

Ex: A=10, B=20

* + - Result = B/A.

document.write(result) gives 2

* + - B%A gives 0

Comparison Operators:

JS supports the following Comparison Operators…

* Equal to (==)
* Not Equal to (! =)
* Greater than (>)
* Less than (<)
* Greater than Or equal to (>=)
* Less than or equal to (<=)
* When used these comparison operators, either ‘true or false’ will be returned to the variable
* While using them it’s better to maintain parenthesis

Ex:

<script type=”text/javascript”>

Var a=10;

Var b=20;

Var result=( a==b )

Document.write(result);

</script>

The above script produces ‘false’ as output

Logical operators:

JS supports the following Logical Operators

* Logical and (&&) - True (1) only if both operands are true (1)
* Logical or ( || ) - True (1) if either of the operands are true (1)
* Logical Not ( ! ) - reverses the logical state of the operand

Bitwise Operators :

Dude.. you need to go through digital logic design to get an idea about this..

* Bitwise and (&)
* Bitwise or (|)
* Bitwise xor (^)
* Bitwise not (~)
* Left shift (<<)
* Right shift (>>)
* Right shift with zero (>>>)

Assignment Operators:

JS supports the following Assignment operators:

* = (Simple assignment )
* += (Add & assignment)
* -= (Subtract & assignment)
* \*= (Multipy & assignment)
* /= (divide & assignment)
* %= (Modulus & assignment)

Same logic applies to Bitwise operators, so they will become <<=, >>=,  
>>=, &=, |= and ^=.

Conditional operator (?:):

Syntax : (Condition) ? True : false

Ex: <script type="text/javascript">

Var a=10;

Var b=20;

Var Result=(a<b)?100:200

document.write(Result);

</script>

Output : 100

Typeof operator:

Typeof operator is a unary operator that’s placed before single operand which is of any type. Typeof operator returns the string indicating the data type of the operand.

It returns the data type of the operand based on the data type of the operand.

|  |  |
| --- | --- |
| Type | String returned by typeof |
| Number | "number" |
| String | "string" |
| Boolean | "boolean" |
| Object | "object" |
| Function | "function" |
| Undefined | "undefined" |
| Null | "object" |

Ex:

<script type="text\javascript">

function telltype()

{

Var a="hey";

Var result = (typeof a == "string" ? "yes":"no")

Document.write(result);

}

</script>

O/p : yes

IF-ELSE

JS supports the following 'if' condition statements :

* If statement
* If-else statement
* If-elseif.. Statement

If else if.. statement :

Ex: <script type="text/javascript">

Var name="praveen";

If(name=="harsha") { alert("great fellow"); }

Else if(name=="dileep") { alert("Golden fellow"); }

Else if(name=="bharfaa") { alert("valuable fellow"); }

Else if(name=="praveen") { alert("Greatest fellow (emo) :D "); }

Else {alert("Dude... provide valid input"); }

</script>

Switch statement

Switch statement behaves like if-else-if.. But its more productive compared to it.

Ex:

<script type="text/javascript">

Var option='s';

Switch(option)

{

Case 'a':document.write("I am praveen");

Break;

Case 'b': document.write("I'm bharfaa");

Break;

Case 's': document.write("U caught me Dude !!");

Break;

Default : document.write("Dude.. Try with another input");

}

Document.write("Hey.. U r done with switch stmt");

</script>

While & do while

While(Entry controlled loop) :

Repeatedly checks till the condition is satisfied.

Ex:

<script type="text/javacript">

Var I =10;

While(I < 10)

{

Document.write(I);

I++;

}

</script>

Do while(Exit controlled loop):

Checks the condition after execution.

<script type="text/javascript">

I=2;

Do

{

Document.write(I);

I++;

}

While(I<10)

</script>

for Loop

For loop in JS is similar to java's for loop.

Syntax: for(initialisation, test condition, iteration statement)

for-in Loop

The for-in loop is used to loop through an object's properties. Each property of an object is assigned to variable till all the object's properties are exhausted.

for(variablename in object)

{

Statements to execute;

}

Loop Control

Break and continue statements :

* Break statement is used to exit the loop at certain point where the break is declared.
* Continue statement tells the control to ignore the 'execution of next lines of code' and continue with the next iteration.(comes out of all loops)