Reference links

About java script

What is js

Its scripting language means, if any issue came in line 20, all previous lines will be executed happily

All browsers will have java script engine, chrome has v8 js engine, hence browser can execute js code

Why js

Its for action purpose – onclick(), auto scroll, to make page dynamic then use js

Notes / syntaxes

1. No need of keeping semicolon at end of each line like in java
2. Using braces to give preference to particular operation, just enclose in braces

document.write(“sum value is ”+ (10+20)) if u don’t enclose in braces then o/p will be “sum value is 1020”

when it sees braces, it will give highest preference to that braces

how js works is, it will see the operator + it will see what is in left and right, it will do that operation in left

1. We can use back tick ~ , double quote “ “ or single quote ‘ ‘ or

Note:- we can write this literals or expressions in double quotes hence prefer to use back tick so that we can use template literal as below

Ex:- u can even create a variable with back tick also

var s1="orini ayya"

var s2=`abba re ----->${s1} <br>`

document.write(`abba re -->${s1} <br>`) // this value will be printed by replacing the s1 in java we use %d here we are using ${var}

output In browser

abba re -->orini ayya

1. Back slash is the escape character for double quotes

console.log("hello \"world\"")

Output:- hello "world"

1. Df

3 ways to print

1. Alert()
2. Document.write(num or any variable)
3. Innerhtml
4. Console.log(2) this will print some data to console

Ways to accept values from browser

1. Using prompt()

var n1= +prompt("enter 1st number")

Variables

1. ways to create variables and

all js variables are case sensitive

var nam=”n1;

var nam =”n2; both these variables are different

java script variables are loosely typed- means we don’t need to mention the type, we can just say var x=”s”;

1. var – which was invented in old versions , with this u can define multiple times

The variables let, const are like java variables, which cant be used before declaring

1. let - new versions
2. const – new versions- these are like final variables u can’t reassign values to that, & it is mandatory to initialise

|  |  |  |
| --- | --- | --- |
| Var (available since starting) | let ( only in new versions this is like a java variable) | const ( only in new versions) |
| 1. Duplicate declaration is accepted,   var x=10  var x=10 Here we defined 2 times still it is accepted | 1. Duplicate declarations are not allowed   let num1=10  let num2=20 //Duplicate declaration of same variable is Not allowed - as already a var is defied with same name  This duplicate is wrong and not possible in let variables (possible only in var variables) |  |
| We can re-assign | We can re-assign | This is a final variable, we can’t re-assign |
| We can use this var before declaration | We **cannot use** this var before declaration | We cant use this variable before declaration |

Data types

Main – number (here all integer, number, float all comes under number only), string, Boolean

Others – undefined, null

Js is dynamically typed language, means u don’t need to specify type explicitly means Integer, String, Double no need to mention these specific types,

just mention var, it will automatically detect the type

Primitive – number (all integer, float, double comes under number), string, Boolean

To check the type of incoming object we should use (typeof(x) operator same like instance of x in java) operator

var x= 23.34;

var x=10; console.log(typeof x) // this will tell us the type of variable ex:- number

Special – undefined, null

Composite- Array, object

|  |  |
| --- | --- |
| We can define strings in Double quotes, single quotes, backticks also | var x="2"; console.log(x); |
|  | var x='1'; console.log(x); |
|  | var x=`2`; console.log(x); |
|  |  |
| Big integer – its like either append with n or write using constructor | var q= 123n; console.log(q); |
|  | var q= BigInt(123n); console.log(q); |
|  |  |
| Undefined  This data type is for un assigned values | By default if u don’t assign js will auto initialise with undefined  Var x; // by default its undefined  Var x= undefined – not recommended use null over this |
|  |  |
| Null | Var x=null; |
|  |  |
| Array | To create array we have to use square braces  //Storing simple types In array  var names = ['one','two','three',null,undefined, true]; console.log(names);  the above is a heterogeneous array  // Storing objects in array    var obj1= {'name': "manideep",'company': "tcs", employeecount : 60000}      var obj2 = {"name":"santhoshi", "company":"IQA", employeecount : 200}     console.log(obj1)     console.log(obj2)     var objTypeArray= [obj1, obj2] |
|  |  |
| Declare a object | In curly braces |

Operators

1. Arithmetic operators +, -,\*,/,%(reminder operator)
2. Binary operators – these will operate on 2 operands ex:- digits ex:- a+b
3. Unary operators – these will operate on single operand ex:- digit 🡪 ++a or a++
4. Logical operators – a&&b (to check if both conditions are true or not), a||b (any condition can be true) , !a

var a=2;

  var b=200;

//These are pre increment or pre decrement operator

document.write(++a) // first it will increment and print

  document.write(--a) // first it will decrement and print

// post increment or decrement operator a++ or a--

Double equal to vs triple equal to

|  |  |
| --- | --- |
| == double equal means it will compare only data not data type  var a=2;    var b=200;      var c= "2";      //Here even though a & c data types are diff (content is same) it just compares content      document.write(a +"=="+c)      document.write(a==c ) | === this triple equal to compares both data and data type // This is called strictly equal to operator  document.write("<br> triple equal to demo")      document.write(a +"==="+c)      //This will check both content and data type      document.write(a===c ) // here since data types are diff this will return false |

1. Assignment operators

|  |  |  |
| --- | --- | --- |
| = | x = y | x = y |
| += | x += y | x = x + y |
| -= | x -= y | x = x - y |
| \*= | x \*= y | x = x \* y |
| /= | x /= y | x = x/y |
| %= | x %= y | x = x % y |
| \*\* | x\*\*= y | x = x \*\* y |
|  |  |  |
|  |  |  |
| &= | x &= y | x = x & y |
| ^= | x^= y | x = x^y |
| |= |  | x = x | y |
| Bit wise operator | | |
| >>= | x >>= y | x = x >> y signed right shift operator  >>> zero fill right shift |
| <<= | x <<= y X | x = x << y zero fill left shift operator |
| & |  | 5 &1 bitwise and operator |
| | |  | 5 | 1 bitwise or operator |
| ~ |  | ~5 not operator |
| ^ |  | ^ xor operator |

6) Ternary operator

  var a=2;

  var b=200;

//   Ternary operator demo

var ternary= a>b? a:b;

document.write("big value is ")

document.write(ternary)

Switch For If- else conditional Statements

|  |  |
| --- | --- |
| Basic **if else** | var num =prompt("enter a positive number")   if (num<0) {      alert("Bro kindly enter positive values only...")   } else {      document.write(" <h1> <br> Thanks u have enterered a positive number -->"+ num +"</h1>")   } |
| Basic **while** – until a condition is false  If u want to execute every time based on a condition | Until u give a positive number this will give alerts keep on  while(num<0) {      alert("Bro kindly enter positive values only...")      num =prompt("enter a positive number")   } |
| If u want to perform a task minimum 1 time max n number of times- then use **do-while**  do {  }  While(condition) | do {          var num=Number(prompt("enter a num to sum"))          sum =sum +num          if(x==0){              x=5;          }          --x;          document.write("<br> entered number is "+ num )      } while (x>0); |
| If  Else if  Else if  …  …  Else  Here also this is possible | let marks=prompt("enter ur marks i will tell d grade");    if(marks>90){     marks="A";    }    else if(marks>80){     marks="B";    }    else if(marks>70){     marks="c";    }    else {     marks="D";    }    document.write(" <h1> <br>your grade is  -->"+ marks +"</h1>") |
| **Switch case**  Break statement is mandatory, else all cases will be executed eventhough all cases have a condition, so keep break statement  Switch(expression) {  Case choice1:  run some code  break;  Case choice2:  run some code:  break;  } | var n1= +prompt("enter 1st number")    var n2=  prompt("enter 2nd number")    var operation = prompt("enter the operator among + or - / or %")    document.write("type of var "+n1 +"is" +(typeof n1))    document.write("type of var "+n2 +"is" +(typeof n2))    switch(operation){     case '+':        alert("The sum value is "+ (n1+n2))        break;     case '-':        alert("The diff is "+ (n1-n2))        break;     case '\*':        alert("the multipled val is "+(n1\*n2))        break ;     default :        alert("The reminder value is "+(n1%n2));    } |
| **For loop** | //Basic for loop  for(let i=0;i<companies.length;i++){      document.write("<br>"+companies[i])  } |
| **For in** loop – **in** means **index** – means when u use this loop u will get index only | //for in loop -- in means index  document.write("<br><br> printing using for in loop <br>")  // this keyword index can be any name  for (index in companies) {      document.write("<br>"+companies[index])  } |
| **For of** loop – this is like **for each** loop In java  When u use this loop u will directly get object  Both these **for in & for of** loop will operate only on **iterable** objects like collection and arrays not on objects, because object is not an iterable it just contain properties & methods | //for of ---- for each loop like in java  document.write("<br> printing using for of loop <br>")  // this keyword element can be any name  for (const element of companies) {      document.write("<br>"+element)  } |
| **Printing object properties using for in loop** | var employee = {"name":"manideep","company":"tata", "vehicle":"bike"}  document.write("printing obj properties using for in loop")  for (key in employee) {      document.write("<br>"+key+"-->"+employee[key])  } |
| **Printing string characters using for in loop** | // printing ele in string using for in loop  let s="wells fargo"  document.write("printing string elements using for in loop")  for(index in s){  document.write("<br> "+s[index])  } |

Type conversion

To check type we should use “typeof” keyword

document.write("type of var "+n1 +"is" +(typeof n1))

document.write("type of var "+n1 +"is" +(typeof(n1)))

|  |  |
| --- | --- |
| String to number – just we need to keep + infront of it | var n1= +prompt("enter 1st number")    var n2=  prompt("enter 2nd number") //Here we didn’t use + so it will be string    var operation = prompt("enter the operator among + or - / or %")    document.write("type of var "+n1 +"is" +(typeof n1))    document.write("type of var "+n2 +"is" +(typeof n2))  output will be  “type of var 12isnumber type of var 2isstring” |
| Any data type 🡪 Number  Number(“any string”) | let x =Number("456")  document.write(x +" is "+ (typeof x) +" <br>") // 456 is number |
| String to number data type (to int value)  parseInt(“any String”)  it takes only the integer part not points and all | let y= parseInt("45F") // This is self intelligent and it will extract 45 num  document.write(y +" is "+ (typeof y)) //45  let z= parseInt("45.02")  document.write(z +" is "+ (typeof z)) //45 |
| String to number (int)  parseFloat(“any float value”)  output will contain points but still data type is float only | let a= parseFloat("11.02")  document.write(a +" is "+ (typeof a) +"<br> ") //11.02 |
| Any data type 🡪 String | //Converting Number to string  b= String(b)  document.write(b +" is "+ (typeof b) +"<br> ")  //Converting boolean to string  b= String(true)  document.write(b +" is "+ (typeof b) +"<br> ") |
| Any data type 🡪 Boolean  Any thing other than zero is true  For zero its is false | let c =0  b= Boolean(c)  document.write(c +" is "+ (typeof b) +" -->"+b+ "<br> ") //false  c= -120  b= Boolean(c)  document.write(c +" is "+ (typeof b) +" -->"+b+ "<br> ")  //true  c= 240  b= Boolean(c)  document.write(c +" is "+ (typeof b) +" -->"+b+ "<br> ") //true  c=1.0  b= Boolean(c)  document.write(c +" is "+ (typeof b) +" -->"+b+ "<br> ") //true  c=`23F`  b= Boolean(c)  document.write(c +" is "+ (typeof b) +" -->"+b+ "<br> ") //true |
|  |  |

Template literal

Template literals means using $

If u want to use template literal then we have to enclose that string in back tick `

1. No need using \n 🡪 In backticks instead of \n, we can directly click enter so that it will print into new line

Ex:-

|  |  |
| --- | --- |
| //Demo for template literal  let profits = '110000cr'  let companyName='Tcs'  //Here this ${val} is called a template literal  //if u are using this literals we have to enclose this string in back tick  document.write(` <br> ${companyName} profits in this year is ${profits} <br>`)  output: Tcs profits in this year is 110000cr | var s1="orini ayya"  var s2=`abba re ----->${s1} <br>`  document.write(s2)  output:  abba re ----->orini ayya |