JAVASCRIPT

Wednesday, October 28, 2020 1:39 PM

In this tutorial we will learn javascript from beginng to advanced and dom with javascript
Lets start our journey

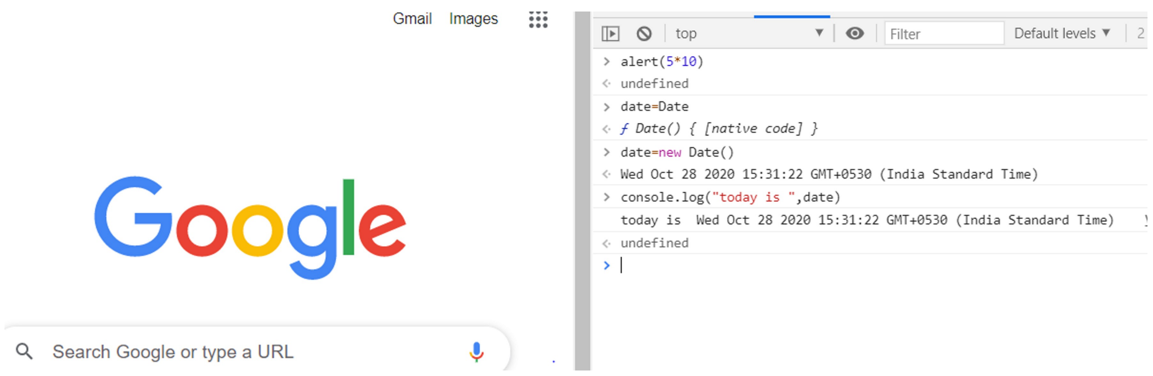
INTRODUCTION:
Front end scripting language
Runs in browser
Interacts with html and css and makes changes in html for the current document

Java and javascript are different
Only java in the name is common apart from that , nothing is common

FOLLOWS ECMASCRIPT2016
This exmascript is an practice which every browser follows to
Refer for compilation of the javascript

Jquery its like abootstrap for javascript

We can goto any browsers console and try few javascripts code snippets
Browsers compile javascript



```
> alert(5*10)
<· undefined
> date=Date
<· ƒ Date() { [native code] }
> date=new Date()
<· Wed Oct 28 2020 15:31:22 GMT+0530 (India Standard Time)
> console.log("today is ",date)
  today is  Wed Oct 28 2020 15:31:22 GMT+0530 (India Standard Time)
<· undefined
> |
```

Adding an inline javascript to html

Browsers renders html+js in top down fashion
The place where we have added the script tag matters

Lets see the example first

```
tml>
    <head>
        <link rel="stylesheet" href="css_for_box.css">
    </head>
<body>
</body>
```

```
<script>
    var date = new Date();
This document refers to current document

Body.innerHtml referes inject the given htmlcode to the current doucments
body


    document.body.innerHTML ="<h1> today is" +date +"</h1>";

</script>
    </html>
```

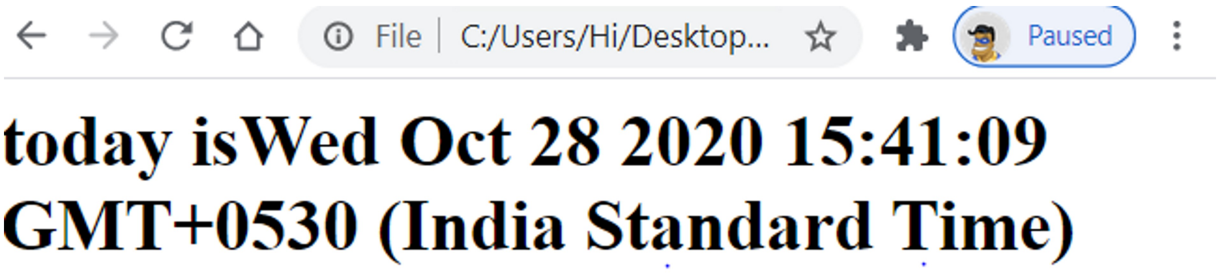
If script placed in head with same body.innerHtml it will throw an error as it didn't find any

Thinking why this script is executing even after we placed after end body tag

If script is placed after end body , as browser previous compiled
Both head and body in top-down fashion , it holds body value
Hence it infuses the extra bit of html to main file

We can place script in body or head or after body according how we would like to render our html

Output :



today is Wed Oct 28 2020 15:41:09 GMT+0530 (India Standard Time)

ADDING EXTERNAL JS TO HTML:

In case if you want to add from external source
<script src="path/filename.js">
</script>

Java script best practices :
1.js is case sentive
2.use camelCase
3.giving whitespace for human readability
4.if possible use semicolon

5// single line // for multiline comments**

==
------DATA TYPES AND VARIBALES IN JAVASCRIPT-----------------------

Var a ,b=10,5;
Sum = a+b;
Note:
var a is local
a means global
Its good to maintain var keyword ,if not it can fetched as a global
Data and will be a security threat

DATA TYPES IN JS:
There are 6 types
1.Numeric --> var a =1 or -1 or 3.112345
2.String ---> var a = "s" or s = " hello \' world "
\ if any extra quoted to be stored in the
3.bool --> var a = True;
4.null var a = null
5.undefined var a ; declare don't assign , leads to undefined
6.Symbol var a = Symbol(id): these are not strings but strings with more benfits
Js serious doenst equate symbols with strings
Symbols are guaranteed to be unique.

```
let id1 = Symbol("id"); let id2 = Symbol("id");
 alert(id1 == id2); // false
```

Typeof(id1) -->symbol
Typeof is type() equivalent in python for js

OPERATORS:
= + - * /
A+(b*c)   follows standard precedence mul first and add next
Supports short hand
A+=1
A++ supports unary operatos
Console.log() -->just to print output output in the browser console

String + number is string

Unmatched operations lead to NaN
Not a number
Since JavaScript treats a dollar sign as a letter, identifiers containing $ are valid variable names:

example
```
var $$$ = "Hello World";
var $ = 2;
```

var $myMoney = 5;

==

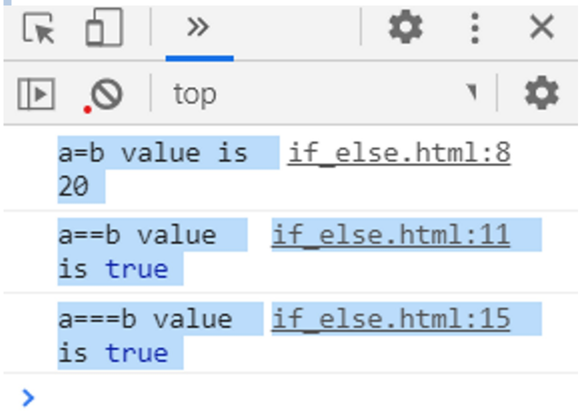
IF ELSE IN JAVASCRIPT:

```
<script>
var a = 20;
var b= "20";
if (a=b){
    console.log("a=b value is",a=b)
}
if (a==b){
    console.log("a==b value is",a==b)
}
if(a===b){
    console.log("a===b value is",a==b)
}
</script>
```

= ASSIGNS THE VALUE
HENCE A POSITIVE VALUE IS ALWAYS TRUE

==
IN PYTHON WORKS IRRESPECTIVE OF STRINGS AND NUMBERS
HENCE COMPARING 55=="55" IS POSSIBLE IN PYTHON
===
THIS IS BEST FOR CHECKING BOTH DATATYPE AND VALUE

SEE THE OUTPUT



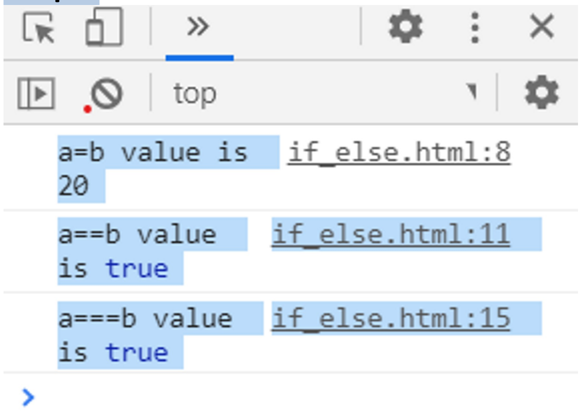
```
a=b value is    if_else.html:8
20
a==b value     if_else.html:11
is true
a===b value    if_else.html:15
is true
>
```

LOGICAL AND/OR AND TERNARY OPERATORS IN JAVASCRIPT:
```
<script>
```

```
var a = 20, b=30;
// lets see logical and and or
if (a==20 && a<=30){
    console.log("a met all conditions under logical and ,both must be true in
 logical and")
}
if (b==20 || b<=30){
    console.log("b met all/any one  conditions under logical or ,both must no
t be true in logical or ||")
}
// lets see the ternary operator in javascript
b>a ? console.log("b>a "):console.log("b<a")
</script>
```

Logical and ==== &&
Logical or ===== ||
Ternary operator == condition ? A:B
If true A ELSE B

Output



```
a=b value is 20                  if_else.html:8
a==b value is true               if_else.html:11
a===b value is true              if_else.html:15
```

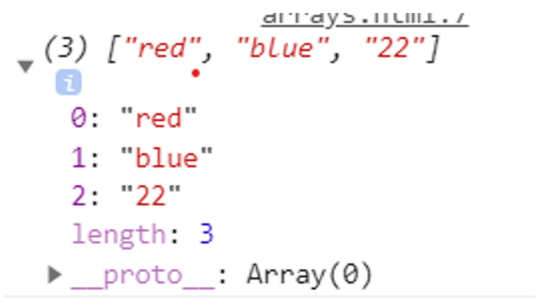
JAVASCRIPT ARRAYS

Js Array is an list equivalent in python
Array is an object in javascript
Each item can be a separate data type

```
<script>
    var pens;
    pens = ["red","blue","22"]
   // array is an list equivalent in python
   // each value can be a seperate data type
   // array is an object in javascript
   console.log(pens)
```

```
</script>
```

Simple array of pens output:



```
arrays.html:7
▼(3) ["red", "blue", "22"]
    0: "red"
    1: "blue"
    2: "22"
    length: 3
  ▶ __proto__: Array(0)
```

PROPERTIES AND METHODS IN ARRAY:-------->

OBJECTS HAVE METHODS AND PROPERTIES
LENGTH IS THE PROPERTY
Array.length
Array.method()
Ex:
Pens.reverse() - reverses the item in the array
Array.shift()-removes the last item in the array

the following code is equipped with more methods

```
<script>
    var pens;
    pens = ["red","blue","green","orange"]
   // array is an list equivalent in python
   // each value can be a seperate data type
   // array is an object in javascript
console.log("hello",pens)
//console.log("reversing the items",pens.reverse())
```

Shift,unshift , push and pop
For deleting and adding items @first and last of the item

```
//pens.shift()
//console.log("shift removes end item of the array",pens)
//pens.unshift("purple","black")
//console.log("adding items @end of arrau using unsfhit",pens)
```

Splice deletes the item @ given position

```
//pens.splice(2,1)
// deletes the items @position and no of items from the position
```

Slice():can create or slice the actual list

```
var new_pens = pens.slice()

//creates a new copy or slice of the array
// pens.indexOf("orange")
```

Joiner joins the list by the given string , just like python

```
joiner = new_pens.join("_--_")
console.log(joiner)

</script>
```

JAVASCIRPT FUNCTIONS AND OBJECTS

The main uses of the functions in javascript are

1.for changing/mopdifying the elements in the html
2.for internal operations

Three types of functions in the js are

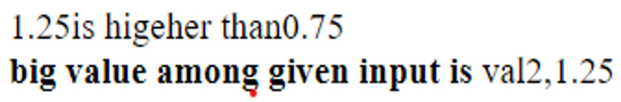
1.named functions
2.anonymus functions
3.invoke functions

Named functions:
General functions like other programming languages
Which accepts parameters and return value
CODE:========>

```
<html>

<body>
</body>
<script>
//function with arguments and return value
function findBigf(a,b){
a>
b ? document.write(a,"is higeher than",b):document.write(b,"is higeher than",
a)
a>b ? big_fraction=["val1",a] :big_fraction=["val2",b]
return big_fraction
}
var val1 =3/4,val2=5/4 ;
big_fraction = new Array()
big_fraction=findBigf(val1,val2)
document.write("<br> <strong>
big value among given input is</strong> ", big_fraction )
</script>
</html>
```

Document.write:
Its an cool function where we can print output on the html screen
Which accepts tags and variable handling functionality
Cool right ? haha

output



```
1.25is higeher than0.75
big value among given input is val2,1.25
```

```
ANONYMUS FUNCTIONS:--->
As like python we can also assign a function definiyion to a variable
Var a = function(x,u ){

Return x
}

IMMEDIATELY INVOKED FUNCTIONS:
```

A variable saving both fuction and return value
For that the structure is

Var a = (function defition(input arguments){
Function logic

})()

```
var x=30 , y=40
var anonymus =  (function(x,y){
 return x**2+y**2
})(x,y)
document.write("<br> <h4>
return value of IMMEDIATELY INVOLED FUCNTIONS<h4> <br><br>",anonymus)
```

-

VARIABLE SCOPE

A variable inside a function - local
A variable without "var" inside a function --global -- be carefull with these
A varibale outside the function global

USE OF VAR AND LET ---> DETAIL EXPLANATION

If we use var for both function level and inner if levels for a same identifier

The value @ inner levels inside the function overrides the value @function level scope
Hence we are using the same identifier with two diffent scopes its best to use "let"
For the inner level variables

```
<html>

<body>
</body>
<script>
    document.write("<h3> var and let explained</h3>")
    function testing_scope_for_var(){

FUNCTION LEVEL VAR A DEFINED

    var a = 20
   document.write("<h5>defined var a with value 20 </h5>")
   document.write("<br>value of var a ",a)

INNER LEVEL VAR A DEFINED

   if(a){
    document.write("<br><h5>
defined var a with value 30 inside function/inside if <h5>")
    var a = 30
    document.write("<br>value of var a ",a)

}
CHECKING THE FINAL OUTPUT OF
FUNCTION LEVEL A ITS BEEN OVVERIDEN BY THE INNER A

document.write("<br>value of var a of function scope is  ",a)
document.write("<h5> as local if value overriding the value outside its scope
 we will use let")
}

function testing_scope_for_let(){

FUNCTION LEVEL B
    var b = 333
   document.write("<h5>defined var b with value 333 </h5>")
   document.write("<br>value of var b ",b)

   if(b){
    document.write("<br><h5>
defined let b with value 329 inside function/inside if </h5>")
INNER  LEVEL B DEFINED WITH LET


    let b = 329
    document.write("<br>value of var b ",b)

}
CHECKING THE FINAL OUTPUT OF
FUNCTION LEVEL B ITS DIFFERENT HERE

document.write("<br>value of var b of function scope is  ",b)
document.write("<h4> hence using let saves our scoping issues and its a must
best practice</h4>")
}

testing_scope_for_var();
testing_scope_for_let();

</script>
</html>
```

OUTPUTS:

var and let explained

defined var a with value 20

value of var a 20

defined var a with value 30 inside function/inside if

value of var a 30
value of var a of function scope is 30

as local if value overriding the value outside its scope we will use let

defined var b with value 333

value of var b 333

defined let b with value 329 inside function/inside if

value of var b 329
value of var a of function scope is 333

hence using let saves our scoping issues and its a must best practice

----
JAVASCRIPT OBJECTS AND CONSTRUCTORS

New keyword for object initialization

Function with arguments can be used as constructor

This.a = a
This keyword for assigning the properties , it acts just like self

```
<script>
//constructor creation
```
The function book acts as the constructor
```
function Book(author,copies,category){
```

This keyword assigns property to object
```
this.author=author
this.category=category
this.copies=copies
this.UpdateCopies = function(){
    return ++this.copies
}
}
//document.write("created a an book object constructor")
```
This
```
var python =new Book("guido van russom",25000,"programming")
document.write(python.author,"<br>",python.copies,"<br>",python.category)
document.write("<br>
updating count afyer you purchased",python.UpdateCopies())
</script>
```

```
guido van russom
25000
programming
updating count afyer you purchased25001
```

JAVASCRIPT CLOUSURES:
These are just like python closures
A FUNCTION INSIDE A FUNCTION WHICH WORKS ON OUTER FUNCTION VARIABLES IS CALLED A CLOSURE
THIS WAY OF CODING SOLVES SO MANY PROBLEMS

LETS CHECK AN EXAMPLE

```
<script>
    document.write("remeber the css course,<br>
we have taken em for calculating sizes of the pixels for compent2 to be two t
imes larger than component 1 ")
    document.write("<br> now we are defining a closure  ")
```

OUTER FUNCTION :

```
function emConverter(pixels){
    base_value = 16 //
```

OUTER FUNCTION LEVEL VARIBALES :

BABY CLOSURSE FUNCTIONS WHICH RETIRNS A VALUE WHEN CALLED:

```
    function converter(){
          return pixels/base_value
      }
     0
```

MAIN FUNCTION RETURNING THE DEFNITION OF THE CLOSURE FUNCTION WITH RETURN VALUE:

```
return converter

}
var small_size = emConverter(16)
var med_size = emConverter(24)
document.write("<br>Small_size value :",small_size()+"em")

document.write("<br> med_size value :",med_size()+"em")
```

HENCE SMALL_SIZE() IS RETURNED FUNCTION WITH A RETURN VALUE FROM MAIN FUNCTION:

```
</script>
```

remeber the css course,
we have taken em for calculating sizes of the pixels for compent2 to be two times larger than component 1
now we are defining a closure
Small_size value :1em
med_size value :1.5em

==============END OF THE JAVASCRIPT BASICS=====================

ADVANCED JAVASCRIPT :
CONTENTS:
1.OBJECTS
2.FUNCTIONS IN DETAIL
3.PROTOTYPES AND IHERITANCE
4.CLASSES
5.ERROR HANDLING
6.PROMISES,ASYNC AND WAIT

OBJECTS IN JAVASCRIPT

OBJECTS EXPLAINED IN DETAIL:

To check key in object property
(key in user) // true/false
FEATURES OF OBJECTS IN JAVASCRIPT:

1. **Properties must be string or symbols**
2. **Values can be of any type**
3. **Accessing can be done by using . Or object[property]**
4. **Del obj.property for object deletion**
5. **Key in obj // true or false for porperty existential checking**
6. **Undefined value for a property leads to false for in checking**
7. **For object iterations**

For var/let key in obj
obj["key"]

Example for the code :
```
<script>
//this is how we write yaml files ,so coool right
const employee ={
    name :"ram"
}
document.write("<br> constant emplyee  name is ",employee.name)
employee.name = "raghu"
document.write("<br> constant emplyee name after change  is ",employee.name)
let user = {
  name: "John",
  age: 30
};
let key = prompt("please add nick_name to the user?");

user.nickname=key
document.write("<br>nick name added<br>",user.nickname)
document.write("<br> lets check a property's existance <br> by == equating wi
th undefined keyword ",(user.nickname==undefined))
document.write("<br> we can fake so that property isnt exist by creating a de
ad property ")
document.write("<br> looping over the object ")
    for (let k in user){
        document.write("<br>",k,"  :",user[k])

    }



</script>
```

```
constant emplyee name is ram
constant emplyee name after change is raghu
nick name added
swa
lets check a property's existance
by == equating with undefined keyword false
we can fake so that property isnt exist by creating a dead property
looping over the object
name :John
age :30
nickname :swa
```

SHALLOW COPY AND DEEP COPY:

Two objects referencing to same property is shallow
Two objects having independent properties is deep
Shallow happens when obj1 is directly initialized with obj2
Deep happens when obj1 is declared empty and assigned properties later

```
<script>
document.write("<h3> lets understand the diffences between shallow and deep c
opy in javascript </h3>")
let user = { name: 'John' };
document.write("<br>user objects name",user.name)
let admin = user;
```

```
document.write("<br>created admin =user lets see admin.name =",admin.name)

document.write("<br> changinf admin.name = peter lets print username")
admin.name = 'Pete';
document.write("<br>user.name is ",user.name);
document.write("<br> if you di1rectly write assign object to another object t
he poperty values gets refered but not independently copied,this is a shallow
 copy")
document.write("<br> checking user==admin and user === admin","   ",user==adm
in,"   ",user===admin)
document.write("<br> as they are both objects with same value == is true,as t
heir objects ids are same due to reference === is also true")
document.write("<br> lets see the how we can create independent data objects"
)
let cloned ={}
for (let key in user){
cloned[key]=user[key]
}
cloned.name = "swaroop"
document.writeln("<br> cloned.name is ",cloned.name," user.name is  ",user.na
me)
document.write("<br>
cheking object status cloned.name===user.name ",cloned.name===user.name )
document.write("<br>we can also do the sam thing using Object.assign")



document.writeln("<h4> end of object's shallow copy,deep copy and assignments
  </h4> " )
</script
```

The above code helps in understanding shallow and deep , which improves our coding and objects initialization much more

Output:

lets understand the diffences between shallow and deep copy in javascript

user objects nameJohn
created admin =user lets see admin.name =John
changinf admin.name = peter lets print username
user.name is Pete
if you di1rectly write assign object to another object the poperty values gets refered but not independently copied,this is a shallow copy
checking user==admin and user === admin true true
as they are both objects with same value == is true,as their objects ids are same due to reference === is also true
lets see the how we can create independent data objects
cloned.name is swaroop user.name is Pete
cheking object status cloned.name===user.name false
we can also do the sam thing using Object.assign

end of object's shallow copy,deep copy and assignments

REACHABILITY AND GARBAGE MANAGEMENT IN JAVASCRTIPT

In js the garbabage collector works on the philosophy called reachability:
When the proprties of the object are being reference or linked with other existing object

-
Some the intrestring points of jascript reachability are as follows

- **Garbage collection is performed automatically. We cannot force or prevent it.**
- **Objects are retained in memory while they are reachable.**
- **Being referenced is not the same as being reachable (from a root): a pack of interlinked objects can become unreachable as a whole.**
- **Refer:**https://javascript.info/garbage-collection

OPTIONAL CHAINING IN JAVASCRIPT:

Optinal chaining is a very faster way to execute the logical comparisions and conditional evaluations in a single go

This is very useful when we have nested obj structures

```
Obj1.property1?.innerProperty
It evaluates whether property is there and returns value if exists returns
Undefined if not

NOTE :THIS IS NOT APPLICABLE IN ARROW FUNCTIONS
```

```
<script>
//lets taste the beauty of optional chaining
document.writeln("<strong>
<center> OPTIONAL CHAINING EXPLAINED IN SIMPLE EXAMPLE </strong></center>")
const animals ={
    dog:{name:"barker"},
    cat:{name:"sophie"}
}
document.writeln("<br> lets have a faster check on the nested properties ")
document.write("<br> lets check whether we have a named cat <br> if yes print
 it",animals.cat?.name )
document.writeln("<br> it so cool feauture else we would have to write hell o
f logical ands and if else cases")
document.writeln("<br> do we have camels in our animal group,lets check anima
l.camels.name ? <br>",animals.camel?.name )
</script>
```

Output:+-

```
        OPTIONAL CHAINING EXPLAINED IN SIMPLE EXAMPLE

lets have a faster check on the nested properties
lets check whether we have a named cat
if yes print itsophie
it so cool feauture else we would have to write hell of logical ands and if else cases
do we have camels in our animal group,lets check animal.camels.name ?
undefined
```

OBJECTS TO PRIMITIVE TYPE CONVERSION:
How js converts obs to primitive data types when operrtions and function calls occur ?

1. **When operations occur ,obj1+obj2 etc, js autocoverts operations accordingly**
2. **Auto conver to string if we are using alert**
3. **Auto convert to number dtype for mathematical**

end of objects

==

Advanced functions in js

Arrow expressions
Structure of arrow expression

Let function name =(arg1,arg2,arg3…,argN) => exrpression;
Ex:

Var sum =(a,b)=> a+b;

Var mul = (x,y)=>x*y;
Multiline arraow is also possible

Val f1=(Args…..)=>{
Lines of code
Retun result

}

A variable which has function expression and return statement with it in a single line mostly

1.RECCURSION EXPLAINED IN JAVASCRIPT

Reccursion is an common topic in every programming lang
A function calling itself
Example :

```
function pow(x, n) { return (n == 1) ? x : (x * pow(x, n - 1)); }
```

Printing linkedlist elements using recursion
This is beautiful and simple program , which traverses using recursion

```
<script>
var list = {
 value :1,
 next :{
          value:2,
          next : {
              value :null
          }
 }
}
//function to display values of linkedlist using recursion
```

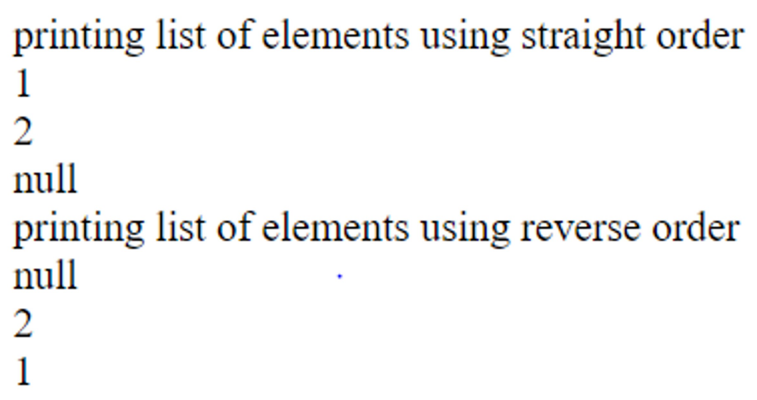
This if condition makes

```
function print_list(list){
document.writeln("<br>",list.value)
          if(list.next){
              print_list(list.next)
          }

}
```

```
Function print_reverse(list){
            if(list.next){
        print_reverser(list.next)

Document.write(list.value)
}
```

The codition breaks @null and prints the current value =2
Then comes back to second call and prints current val 1
Then comes back to the first call and prints current val 0
Then exits the control

```
</script>
```



```
printing list of elements using straight order
1
2
null
printing list of elements using reverse order
null
2
1
```

SPREAD AND REST PARAMETES IN JS
SPREAD - For expanding array
Var a [1,2,3]
Var b =[4,5,6]
Var c =[..a,22]
Now c has [1,2,3,22]

Rest :
If we use the same technique for arguments in a function
Then its called rest
Array functions don’t have rest

Function hello(…string){
String[0]
String[1]

}
Note :
Settimeout --> run once after wait period
setInterval->run repeatedly for short intervals of time

ARROW FUNCTIONS
~ Arrow functions allows us to write shorter syntax functions

```
<script>
 document.write("<center><h3> lets discuss ARROW FUNCTIONS </h3></center>")
hello = ()=> {
    return " a hello from arrow functions"
}
//arrow functions return by default
name=()=> "my name is SWAROOP LEKHARAJU "
document.write("<br>",name)
// arrow functions with paranthesis
age =(x)=> "my age is "+x
let x =25
document.write("<br>",age(x))

document.write("<br> in normal functions this represnts the independent objec
t, in arrow this represents the owner")
let resturant1 ={
   spcl:"pasta",
   beverage : "ice peach tea" ,
   // inner method for normal function
   order :function(){
     return'i will have 1 '+this.spcl

   }
}
document.write("<br>",resturant1.order())
document.write("<br> lets try the same with arrow functions")
let resturant2 ={
spcl:"pizza",
order:()=> 'i will have 1 '+this.spcl

}
//here arrow function will say undefined ,bcz its not binded to the main func
tion
document.write("<br>", resturant2.order())
</script>
```