Javascript

\*\*All thing about JavaScript

https://overapi.com/javascript

**\*\*console.log**

**console.log("console log is here");**

**console.log(6.789);**

**console.log("my fav number "+ 96);**

**console.log(9+9);**

**console.log("9"+9);**

**\*\*Variables**

***var* name="Rasel";**

***var* age=20;**

**console.log(name+" knows Javascript.");**

**console.log("His age is only "+age);**

**console.log(name+" is creating a javascript course for all");**

**console.log("But his age is only "+age);**

**\*\*Datatype**

**1.Primitive: Primitive data types are defined by javascript**

* + **Number => 25,25.5;**
* ***var* integer= 1451;**
* ***var* float= 3.14;**
* ***var* n1= *Number*(45);**
* ***var* nn= *Number*("45");**
* **console.log(nn+n1);**
* ***var* num = 555.6;**
* **console.log(*Number*.parseFloat(num));**
* **console.log(*Number*.parseInt(num));**
* **console.log(*Number*.MAX\_VALUE);**
* **console.log(*Number*.MIN\_SAFE\_INTEGER);**
* **console.log(1/0); //return infinity**
* **console.log('abc'/10); //return NaN**

**Hexadecimal = 16 base number system,Octal = 8 base number system**

***var* hex= 0xff;**

**console.log(hex);**

***var* oct=0756;**

**console.log(oct)**

* + **String => “This is Strign”;**
* ***var* str='string';**
* ***var* str2="string";**
* ***var* str3=`string`; //backtick**
* ***var* str4=*String*("anything");**
* ***var* str5=*String*(156);**
* ***var* str6=*String*(16.625);**
* **console.log(str,str2,str3,str4,str5,str6);**
  + **Blooean => true or false;**
* ***var* b1 = true;**
* ***var* b2 = false;**
* ***var* b3=*Boolean*(true);**
* ***var* b4=*Boolean*(false);**
* ***var* b5=*Boolean*(0);**
* ***var* b6=*Boolean*(1);**
* **console.log(b1,b2,b3,b4,b5,b6);**
  + **Undefine=> return false value**
  + **Null => return false value**
* ***var* abc;**
* ***var* xyz=null;**
* **console.log(abc,xyz)**

**2.Object: Object Types are user Defined data type**

**\*\*Type Conversion**

***var* str='100';**

***var* n=10;**

**console.log(str\*n); //automated conversion**

**console.log(str+n);**

**console.log(*Number*(str))**

**console.log(*Number*.parseFloat(str));**

**console.log(*Number*.parseInt(str));**

***var* num=10;**

**console.log(num);**

**console.log(num.toString());**

**console.log(*Number*(1/0));**

**console.log(*Number*(Infinity));**

**console.log(Infinity.toString());**

**console.log(*Boolean*(Infinity)); //return True**

**console.log(*Boolean*(-Infinity)); //return True**

**console.log(true);**

***var* x=true;**

**console.log(x.toString()); //String**

**\*\*Falsy Values :**

* + - **‘ ’**
    - **0**
    - **null**
    - **undefined**
    - **NaN**
* **console.log(*Boolean*('')); // false**
* **console.log(*Boolean*("Rasel Hossain")); //true**
* **console.log(*Boolean*(null)); // false**
* **console.log(*Boolean*(undefined)); // false**
* **console.log(*Boolean*(0)); // false**
* **console.log(*Boolean*(-0)); //false**
* **console.log(*Boolean*(451)); //true**
* **console.log(*Boolean*(-88)); //ture**

falsy : https://developer.mozilla.org/en-US/docs/Glossary/Falsy

truthy : https://developer.mozilla.org/en-US/docs/Glossary/Truthy

\*\* Constructor

1.String 2.Number 3.Boolean

**new string(“string”) => it’s return an array**

Operators

\*\*Arithmetic Operator:

+ ,- ,\* ,/ ,% ,++ ,--

**\*\*increment and dicrement**

**console.log(++a) //first incriment**

**console.log(a++); //second time incriment**

**console.log(a) //second time incriment result**

**console.log(--a) //first dicrement**

**console.log(a--) //second time dicrement**

**console.log(a) //second time dicrement**

\*\* Assignment Operator:

= , +=, -=, /=, \*=, a%b

a+=b (a=a+b)

a-=b(a=a-b)

\*\*Comparison Operators:

Comparison operators always return Boolean value

== , === , != ,!== ,> , < , >= , <=

Comparison :

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions\_and\_Operators

***var* a=10;**

***var* b=20;**

**console.log(a==b); //false**

**console.log(a!=b); // true**

**console.log(a>b); //false**

**console.log(a<b) //true**

**console.log(a >= b) //false**

**console.log(a <= b) //true**

**== ===**

***var* c='50';**

***var* d=50;**

**console.log(c==d);**

**console.log(c===d);**

**console.log(c!==d);**

\*\*Logical Operators:

OR = ||

AND = &&

NOT = !

\*\* Bitwise Operators:

OR = | (Binary or)

AND = & (Binary and)

NOT = ! (Binary not)

\*\*typeof operator:

**console.log(typeof(“Strin”) ;**

**\*\*Statement**

**console.log("Hello World");**

***var* str='string';**

***var* number=10+(30/5)\*5; // ,() , / , \* , + , -**

***var* a=10;**

***var* b=20;**

***var* c= a>b;**

Math() function

**console.log(Math.E);**

**console.log(Math.PI);**

**console.log(Math.abs(-999668));**

**console.log(Math.floor(5.9));**

**console.log(Math.ceil(5.1));**

**console.log(Math.round(5.5));**

**console.log(Math.max(6,97,0,80,100));**

**console.log(Math.min(6,97,0,80,100));**

**console.log(Math.pow(2,2));**

**console.log(Math.sqrt(36));**

**console.log(Math.round(Math.random()\*50))**

max value using array:

**var arr = [1, 2, 3,5,9,0];**

**var max = Math.max(...arr);**

**console.log(`${max}`);**

***// Using Function***

**var maxValue=function(arr){**

**return Math.max(...arr);**

**}**

**console.log(maxValue(arr))**

new date()

***var* date=new *Date*();**

**console.log(date.toDateString());**

**console.log(date.toTimeString());**

**console.log(date.toLocaleString()); //Bangladesh time**

**console.log(date.toLocaleDateString());**

**console.log(date.toLocaleTimeString());**

**console.log(date.getFullYear());**

**console.log(date.getMonth());**

**console.log(date.getDate());**

**console.log(date.getHours());**

**console.log(date.getMinutes());**

**console.log(date.getSeconds());**

**console.log(date.getMilliseconds()); //1000 millisecond =1 second**

Logic and Condition

***var* number=5;**

**if(number % 2 === 0){**

**console.log("Number is Even");**

**}if((number % 2 ===1) && (number % 2 !== 0)){**

**console.log("number is odd")**

**}**

***var* a=10;**

***var* b=20;**

**if(a>b){**

**console.log(a +" is greater than "+b);**

**}else{**

**console.log(b +" is greater than "+ a);**

**}**

\*\*7 days name count with if else and switch

***var* date=new *Date*();**

***var* toDay=date.getDay();**

**if(toDay == 0){**

**console.log("SunDay");**

**}else if(toDay == 1){**

**console.log("Monday");**

**}else if(toDay==2){**

**console.log("Tuesday");**

**}else if(toDay==3){**

**console.log("Wednesday");**

**}else if(toDay==4){**

**console.log("Thursday");**

**}else if(toDay==5){**

**console.log("Friday");**

**}else if(toDay==6){**

**console.log("Saturday");**

**}**

**Switch**

**switch(toDay){**

**case 0:{**

**console.log("SunDay");**

**break;**

**}**

**case 1:{**

**console.log("Monday");**

**break;**

**}**

**case 2:{**

**console.log("Tuesday");**

**break;**

**}**

**case 3:{**

**console.log("Wednesday");**

**break;**

**}case 4:{**

**console.log("Thursday");**

**break;**

**}case 5:{**

**console.log("Friday");**

**break;**

**}**

**default:{**

**console.log("Saturday");**

**break;**

**}**

**}**

\*\* if else with Logical operators:

***var* a=10;**

***var* b=20;**

***var* c=30;**

***var* d=40;**

**if(a>b && a>c && a>d){**

**console.log("a is big number");**

**}else if((b>a) && (b>c) && (b>d)){**

**console.log("D is Big Number");**

**}else if((c>a) && (c>b) && (c>d)){**

**console.log("C is Big Number");**

**}else if((d>a) && (d>b) && (d>c)){**

**console.log("D is Big Number");**

**}**

Ternary Operator

***var* n=11;**

***var* str='';**

**if(n%2==0){**

**str='Even';**

**}else{**

**str="Odd";**

**}**

**console.log(str);**

\*\*Ternary operator example:

***var* result=n%2==0 ? "Even" : "ODD";**

**console.log(result);**

**// condition ? true side : false side**

How to use AND OR Shorthand

***var* name="";**

***var* fullname='';**

**if(name.length==0){**

**fullname="HM Nayem";**

**}else{**

**fullname=name;**

**}**

**console.log(fullname);**

\*\*shorthand

***var* name="";**

***var* fullname=name || 'Rasel Hossain';**

**console.log(fullname); //pring Rasel Hossain**

***var* name="HM Nayem";**

***var* fullname=name || 'Rasel Hossain';**

**console.log(fullname); //pring HM Nayem**

***var* isOK='';**

***var* result=isOK || "left side";**

**console.log(result)**

***var* isOk=true;**

**isOk && console.log("Everything is ok");**

***var* isOk=(5==5);**

**isOk && console.log("Everything is ok");**

***var* isOk=false;**

**isOk && console.log("Everything is ok");**

Learn more shorthand : https://www.sitepoint.com/shorthand-javascript-techniques/

Loop

For loop

While loop

Do while loop

Foreach loop

For in loop

For of loop

\*\* For loop

* Print Your name 10000 times

**for(*var* i=1;i<=10000;i++){**

**console.log(i+"rasel ")**

**}**

**for(*var* i=1;i==false;i++){**

**console.log(i+"rasel ")**

**}**

**for(*var* i=1;i<=100;i+=2){**

**console.log(i+"rasel ")**

**}**

**for(*var* i=1;i<=100;i--){**

**console.log(i+"rasel ")**

**}**

\*\*1 to 100 Odd and Even Number print

\*\*1 to 100 sum

\*\*1 to 100 even number count

***var* count=0;**

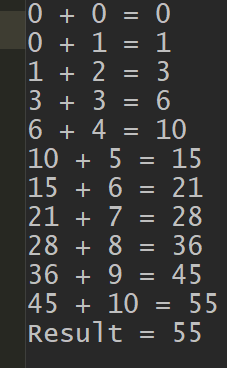
**for(*var* i=2;i<=100;i+=2){**

**console.log(i);**

**++count;**

**}**

**console.log("value "+count)**

\*\*Do This => 0 to 10 sum and print like this

Code :

***var* sum=0;**

**for(*var* i=0;i<=10;i++){**

**console.log(sum + " + "+ i + " = " + (sum+i));**

**sum+=i;**

**}**

**console.log("Result = "+sum);**

\*\*While Loop

var initialization=1;

While(condition){

operation;

update;

}

1. 1 to 100 number print

***var* isRuning=true;**

**while(isRuning){**

***var* rand=Math.floor(Math.random() \*10+1);**

**if(rand==9){**

**console.log("winner winner chicken dinner");**

**isRuning=false;**

**}else{**

**console.log("You have gor "+rand)**

**}**

**}**

\*\*do while loop

***var* isRuning=false;**

**do{**

**console.log('i am runing')**

**}while(isRuning);**

***var* isRuning=false;**

**do{**

**console.log("First I am Runing");**

**}while(isRuning);**

\*\*Nested Loop

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

1 2 3 4 5 6

**for(*var* i=1;i<=6;i++){**

***var* result="";**

**for(*var* j=1;j<=i;j++){**

**result+=j;**

**}**

**console.log(result);**

**}**

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

\* \* \* \* \* \*

\* \* \* \* \* \* \*

**for(*var* i=1;i<=6;i++){**

***var* result="";**

**for(*var* j=1;j<=i;j++){**

**result+="\* ";**

**}**

**console.log(result);**

**}**

\* \* \* \* \* \*

\* \* \* \* \* \*

\* \* \* \* \* \*

\* \* \* \* \* \*

\* \* \* \* \* \*

\* \* \* \* \* \*

**for(*var* i=1;i<=6;i++){**

***var* result="";**

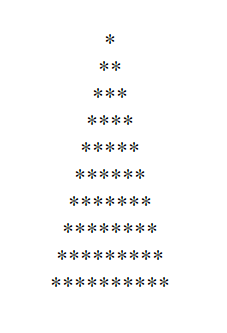
**for(*var* j=1;j<=6;j++){**

**result+="\* ";**

**}**

**console.log(result);**

**}**

do this in browser

**document.write("<center>");**

**for(*var* i=1;i<=10;i++){**

**for(j=1;j<=i;j++){**

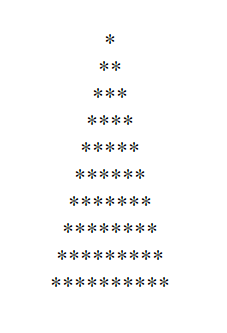
**document.write("\*");**

**}**

**document.write("<br>");**

**}**

**document.write("</center>");**

browser print

**for(*var* i=1;i<=10;i++){**

**for(*var* k=1;k<=(10-i);k++){**

**document.write("&nbsp");**

**}**

**for(j=1;j<=i;j++){**

**document.write("\*");**

**}**

**document.write("<br>");**

**}**

Print in browser

**for(*var* i=1;i<=5;i++){**

**for(*var* k=1;k<=(5-i);k++){**

**document.write("&nbsp");**

**}**

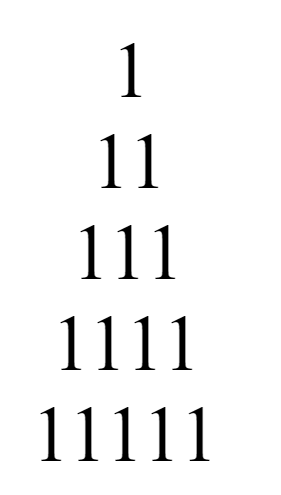
**for(*var* j=1;j<=i;j++){**

**document.write(i);**

**}**

**document.write("<br>");**

**}**



**for(*var* i=1;i<=5;i++){**

**for(*var* k=1;k<=(5-i);k++){**

**document.write("&nbsp");**

**}**

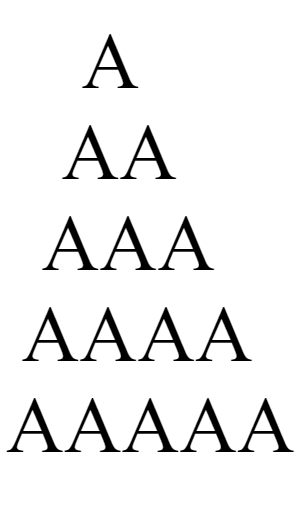
**for(*var* j=1;j<=i;j++){**

**document.write("1");**

**}**

**document.write("<br>");**

**}**



**for(*var* i=1;i<=5;i++){**

**for(*var* k=1;k<=(5-i);k++){**

**document.write("&nbsp");**

**}**

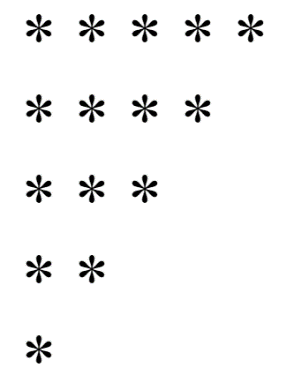
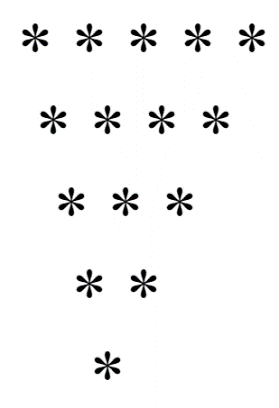
**for(*var* j=1;j<=i;j++){**

**document.write("A");**

**}**

**document.write("<br>");**

**}**



**for(*var* i=5;i>=1;i--){**

**for(*var* j=1;j<=i;j++){**

**document.write("\* ");**

**}**

**document.write("<br>");**

**}**

**for(*var* i=5;i>=1;i--){**

**for(k=1;k<=(5-i);k++){**

**document.write("&nbsp");**

**}**

**for(*var* j=1;j<=i;j++){**

**document.write("\* ");**

**}**

**document.write("<br>");**

**}**

\*\*Break

**for(*var* i=1;i<=10;i++){**

**if(i%5==0){**

**break;**

**}else{**

**console.log(i);**

**}**

**}**

\*\*continue

**for(*var* i=1;i<=10;i++){**

**if(i==3 || i==7){**

**continue;**

**}else{**

**console.log(i);**

**}**

**}**

\*\*infinity Loop:

Do this while loop program using for loop

***var* isRuning=true;**

**while(isRuning){**

***var* rand=Math.floor(Math.random()\*10+1);**

**if(rand==9){**

**console.log("Winner Winner Chicken Dinner");**

**isRuning=false;**

**}else{**

**console.log("You have got "+rand);**

**}**

**}**

**for(; ;){**

***var* rand=Math.floor(Math.random()\*10+1);**

**if(rand==9){**

**console.log("Winner Winner Chicken Dinner");**

**break;**

**}else{**

**console.log("You have got "+rand);**

**}**

**}**

\*\* All About String

***var* str="String"; //String Literal**

***var* str2=*String*("String"); //String Constructor**

***var* n=10;**

***var* str=n;**

**console.log(str); //this is return Number**

***var* n=10;**

***var* str=n+"";**

**console.log(str); //this is return String**

***var* n=10;**

***var* str=*String*(n);**

**console.log(str); //this is return String**

***var* n=10;**

***var* str=n.toString();**

**console.log(str); //this is return String**

Escape Notation

‘ “\ \” ‘

***var* str="This is a \"String\"";**

n\

***var* str="This is a \nString";**

\t

***var* str="This is a \tString";**

\\

***var* str="This is a \\String";**

\*\*String Comparison

Small letter is big , capital letter is small

a is big , A is small a>A

a is small , z is big a<z

abc > bcd = false

abc < bcd = true

abz > bcd = true

abz < zaa = true

aaa === aaa = true

aaaZ > aaaz = false

aaz > aaaZ = true

“001” == 1 = true

“001” === 1 = false

\*\* String Method / Function

.concat()

***var* a = "i am ";**

***var* b="Rasel";**

***var* c=a.concat(b);**

**console.log(c)**

.substr(start number,end number)

Or .substr(start number)

***var* a = "i am ";**

***var* b="Rasel";**

***var* c=a.concat(b);**

***var* d=c.substr(5)**

**console.log(d)**

.charAt(number)

***var* a = "i am ";**

***var* b="Rasel";**

***var* c=a.concat(b);**

***var* d=c.charAt(5)**

**console.log(d)**

.startsWith(“letter”)

***var* a = "i am ";**

***var* b="Rasel";**

***var* c=a.concat(b);**

***var* d=c.startsWith("i")**

**console.log(d)**

.endsWith(“letter”)

***var* a = "i am ";**

***var* b="Rasel";**

***var* c=a.concat(b);**

***var* d=c.endsWith("Rasel")**

**console.log(d)**

.toLowerCase();

.toUpperCase();

.trim();

.split(‘ ’)



\*\* String Length:

***var* str="raselofficial.com";**

***var* length=0;**

**while(true){**

**if(str.charAt(length)==""){**

**break;**

**}else{**

**length++;**

**}**

**}**

**console.log(length)**

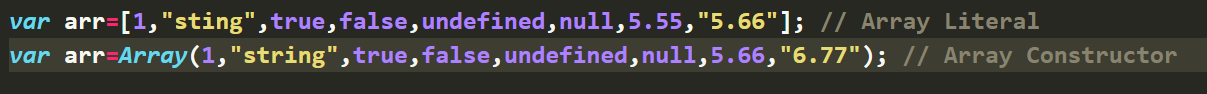
another way

***var* str="raselofficial.com";**

***var* length=str.length;**

**console.log(length);**

Array



***var* arr=[1,2,3,4,5];**

**console.log(arr); //pring all array**

**arr[5]=6; //data insert in index 5**

**arr[10]= 11; //data insert in index 10**

**arr[1]=11; //replace index 1 data;**

**console.log(arr); //pring all array**

**console.log( arr[4] ) //print 4 number index value**

**console.log(arr.length);**

\*\*Array Traversing

***var* arr=[1,2,3,4,5,6,7,8,9,10];**

**for(*var* i=0;i<=(arr.length-1);i++){**

**console.log(arr[i]);**

**}**

\*\* + 2 every array value

***var* arr=[1,2,3,4,5,6,7,8,9,10];**

**for(*var* i=0;i<=(arr.length-1);i++){**

**arr[i]+=2; //arr[i]=arr[i]+2;**

**}**

**console.log(arr)**

\*\* add array all value

***var* arr=[1,2,3,4,5,6,7,8,9,10];**

***var* sum=0;**

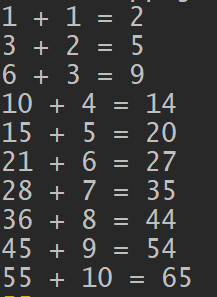
**for(*var* i=0;i<=(arr.length-1);i++){**

**sum+=arr[i]; //sum=sum+arr[i];**

**}**

**console.log(sum)**

\*\* add all array value



***var* arr=[1,2,3,4,5,6,7,8,9,10];**

***var* sum=0;**

**for(*var* i=0;i<arr.length;i++){**

**sum+=arr[i];**

**console.log(sum+" + "+arr[i]+" = "+(sum+arr[i]))**

**}**

\*\* print even number from array

***var* arr=[1,2,3,4,5,6,7,8,9,10];**

***var* sum=0;**

**for(*var* i=0;i<arr.length;i++){**

**if(arr[i]%2==0){**

**console.log(arr[i]);**

**}**

**}**

\*\*print first even number from

***var* arr=[1,2,3,4,5,6,7,8,9,10];**

***var* sum=0;**

**for(*var* i=0;i<arr.length;i++){**

**if(arr[i]%2==0){**

**console.log(arr[i]);**

**break;**

**}**

**}**

Insert Remove and Replace Array Data

***var* arr=[1,2,3,4,5,6,7,8,9,10];**

**// insert 9 to index 3**

**arr[3]=9;**

**// insert data to last index**

**arr.push(90);**

**//insert data to first index**

**arr.unshift(90);**

**//insert data specific place/index**

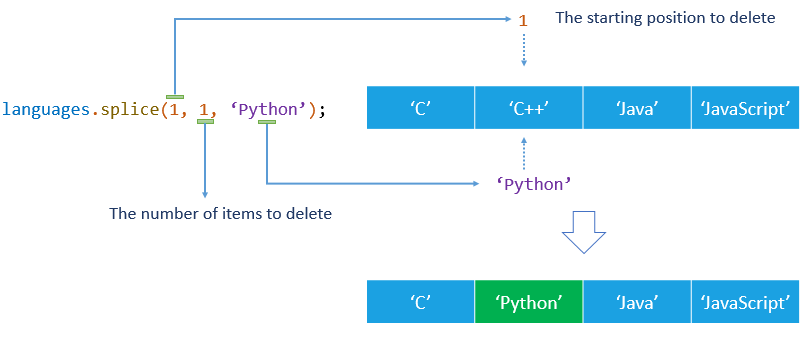
**arr.splice(3,2,"rasel");**

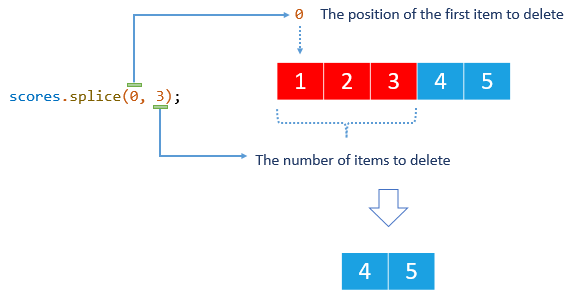
**console.log(arr)**

**//insert data specific place**

**arr.splice(4,0,"rasel");**

**console.log(arr)**







Do this=>

Do this=>

**arr[3]= undefined;**

\*\* Delete last value

**arr.pop();**

\*\*Delete First value

**arr.shift();**

\*\*Data Search From Array

***let* arr=[1,6,23,37,347,8346,23,6,77,856];**

***var* usrData=77;**

***var* isFound=false;**

**for(*var* i=0;i<arr.length;i++){**

**if(arr[i]==usrData){**

**console.log("Data Found!");**

**isFound=true;**

**break;**

**}**

**}**

**if(!isFound){**

**console.log("Data Not Found!");**

**}**

\*\*Multidimensional Array

***var* arr2;**

***var* arr=[**

**[1,2,3,4,5,6]**

**]**

\*2 Dimensional Array

***var* arr=[**

**[],**

**[],**

**[]**

**]**

\*3 Dimensional Array

***var* arr=[**

**[**

**[],**

**[],**

**[]**

**],**

**[**

**[],**

**[],**

**[]**

**],**

**[**

**[],**

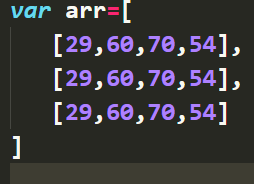
**[],**

**[]**

**]**

**]**

\*\*travers this array



***var* arr=[**

**[29,60,70,54],**

**[29,60,70,54],**

**[29,60,70,54]**

**]**

**for(*var* i=0;i<arr.length;i++){**

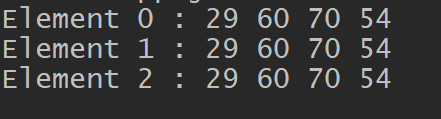
**for(*var* j=0;j<arr[i].length;j++){**

**console.log(arr[i][j]);**

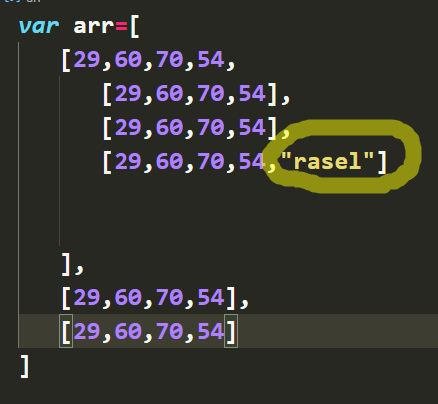
**}**

**}**

\*Print Like This



\*\* Find “rasel” value from this array using Loop



**\*\*print Start**

**var arr=[**

**["\*"],**

**["\*","\*"],**

**["\*","\*","\*"],**

**["\*","\*","\*","\*"],**

**["\*","\*","\*","\*","\*"],**

**["\*","\*","\*","\*","\*","\*"],**

**["\*","\*","\*","\*","\*","\*","\*"],**

**];**

**for(var i=0;i<arr.length;i++){**

**var result ="";**

**for(var j=0;j<arr[i].length;j++){**

**result+=arr[i][j];**

**}**

**console.log(`${result}`);**

**}**

\*\* insert arr2 data in arr

**var arr=[1,2,3,4,5,6,7];**

**var arr2=[8,9,10,11,12,13,14];**

**for(var i=0;i<arr.length;i++){**

**arr[i]=arr2[i]**

**}**

**console.log(arr)**

\*\*Reverse An Array

***var* arr=[1,2,3,4,5,6,7]**

**for(*var* i=0;i<(arr.length/2);i++){**

***var* temp=arr[i];**

**arr[i]=arr[arr.length-1 -i];**

**arr[arr.length-1-i]=temp;**

**}**

**console.log(arr);**

\*\*Another way (But Wrong)

***var* arr=[1,2,3,4,5,6,7,8];**

**for(*var* i=0;i<(arr[arr.length/2]);i++){**

**arr[i]=arr[arr.length-1-i];**

**arr[arr.length-1-i]=arr[i];**

**}**

**console.log(arr);**

\*Do this to this

Arr.reverse();

***var* arr=[1,2,3,4,5,6,7]**

**arr.reverse();**

**console.log(arr)**

arr.jone(“separator”);

***var* arr=[4,5,9,6];**

**console.log(arr.join("separator"))**

***var* arr=[4,5,9,6];**

**console.log(arr.join(" | "))**

\*\*All Thing about JavaScript

<https://overapi.com/javascript>

\*arr.fill(value);

***var* arr=[4,5,9,6];**

**arr.fill(0);**

**arr.fill("0");**

**arr.fill(true);**

**console.log(arr)**

\*\*arr.concat(seconde array)

***var* arr=[4,5,9,6];**

***var* arr2=[5,60,80,21];**

**arr=arr.concat(arr2)**

**console.log(arr)**

\*Array.isArray(array)

***var* arr=[4,5,9,6];**

***var* arr2=[5,60,80,21];**

**console.log(*Array*.isArray(arr))**

\*Array.from(arr1);

***var* arr=[4,5,9,6];**

***var* arr2=*Array*.from(arr);**

**console.log(arr)**

Object

\*Declare object

***var* obj={} //object literal**

***var* obj={}**

**console.log(typeof(obj))**

***var* obj={}**

**obj.x = 10**

**console.log(obj)**

***var* point={**

**x:10,**

**y:60,**

**j:80**

**}**

**console.log(point)**

\*print object value

**console.log(point.y)**

\*replace a data in “y” property

**point.y=10;**

\*add property value

***var* point={**

**x:10,**

**y:60,**

**j:80**

**}**

**point.y=10;**

**point.z=30;**

**console.log(point)**

***var* obj=*Object*() //constractor**

**obj.a=10,**

**obj.b=30**

**console.log(obj)**

\*Another way to declare object constractor

***var* obj=new *Object*() //constractor**

**obj.a=10,**

**obj.b=30**

**console.log(obj)**

\*\*Accessing Object

\*\*Dot Notation

***var* point={**

**x:10,**

**y:20,**

**z:15**

**}**

**// ----------- Dot Notation --------**

**// print x property value**

**console.log(point.x)**

**// print y property vlaue**

**console.log(point.y)**

**// add(+) x and y property value**

**console.log(point.x + point.y)**

\*\*Array Notation

***var* point={**

**x:10,**

**y:20,**

**z:15**

**}**

**// ----------- Array Notation --------**

**// print x property value**

**console.log(point['x'])**

**// print y property vlaue**

**console.log(point['y'])**

**// add(+) x and y property value**

**console.log(point['x'] + point['y'])**

Setting Object Properties

***var* point={**

**x:10,**

**y:20,**

**z:15**

**}**

**point['b']=40;**

**console.log(point)**

***var* point={**

**x:10,**

**y:20,**

**z:15**

**}**

***var* B=['b'];**

**point[B]=40;**

**console.log(point)**

***var* point={**

**x:10,**

**y:20,**

**z:15**

**}**

***var* user=*String*(prompt("Enter Your Name"));**

**point[user]=60;**

**console.log(point)**

\*Update value

***var* point={**

**x:10,**

**y:20,**

**z:15**

**}**

**// update x property vlaue**

**point['x']=20;**

**console.log(point)**

\*undefined

***var* point={**

**x:10,**

**y:20,**

**z:15**

**}**

**// update x property vlaue**

**point['x']=20;**

**console.log(point.d) //undefiend**

\*\*Remove Object Properties

delete objectName.property

***var* point={**

**x:10,**

**y:20,**

**z:15**

**}**

**delete point.x; //delete operator**

**console.log(point)**

Comparing Two Objects

\*tow object compare

***var* obj1={**

**a:10,**

**b:20**

**}**

***var* obj2={**

**a:10,**

**b:20**

**}**

**console.log(obj1===obj2) //output false**

**// you must have to compare object property not object**

**// Right way to compar object**

**if(obj1.a===obj2.a){**

**console.log(true)**

**}else{**

**console.log(false)**

**}**

***var* obj1={**

**a:10,**

**b:20**

**}**

***var* obj2={**

**a:10,**

**b:20**

**}**

**// you must have to compare object property not object**

**// Right way to compare object**

**if(obj1.a===obj2.a && obj1.b===obj2.b){**

**console.log(true)**

**}else{**

**console.log(false)**

**}**

\*\*convert object to String

 to



\*\*now we can compare tow object directly

***var* obj1={**

**a:10,**

**b:20**

**}**

***var* obj2={**

**a:10,**

**b:20**

**}**

**obj1=JSON.stringify(obj1);**

**obj2=JSON.stringify(obj2)**

**console.log(obj1===obj2) //Output true**

iterate object property

‘x’ in objectName operator:

***var* obj={**

**x:40,**

**y:60,**

**z:75**

**}**

**console.log( 'x' in obj ) //output true**

**console.log( 'p' in obj ) //output false**

Traversing Object

Note: the problem for travers object

* Doesn’t have any starting number
* Object property name is different
* Can’t increment or decrement

Solution: we can use for in loop for traversing objects

for ( var i in objectName ){

operation

}

***var* obj={**

**x:40,**

**y:60,**

**z:75**

**}**

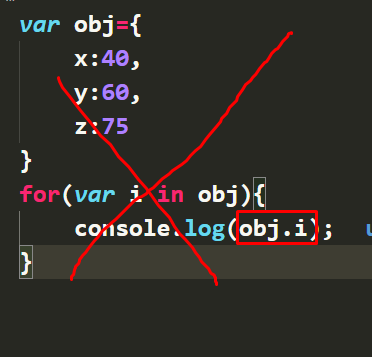
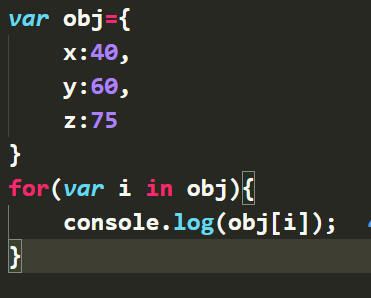
**for(*var* i in obj){**

**console.log(i);**

**}**

\*\*print property value using for in loop

this is wrong way because we can’t access object property using variable with dot(.) notation if I want to access object variable than I have to use Array[‘ ‘] notation

dot(.) notation Array [ ] notation

***var* obj={**

**x:40,**

**y:60,**

**z:75**

**}**

**for(*var* i in obj){**

**console.log(obj[i]);**

**}**

***var* obj={**

**x:40,**

**y:60,**

**z:75**

**}**

**for(*var* i in obj){**

**console.log(`${i} : ${obj[i]}`);**

**}**

Object Method

Object.keys(objectName) :

***var* obj={**

**x:40,**

**y:60,**

**z:75**

**}**

***var* arr=*Object*.keys(obj); //print object property it's return an array**

***var* value=*Object*.values(obj) //print object property value it's return an array**

***var* entries=*Object*.entries(obj); // return  multidimensonal Array**

**console.log(arr)**

**console.log(value)**

**console.log(entries)**

\*\*Create Object from Existing Array

***var* obj1={**

**x:40,**

**y:60,**

**z:75**

**}**

***var* obj2=obj; //we can do it but there are a problem,if i change any value of obj2 than the value also change of obj1 object**

**obj2.x=50;**

**console.log(obj1)**

**console.log(obj2)**

the solution is: Object.assign({},obj);

***var* obj2=*Object*.assign({},obj)**

***const* target = { a: 1, b: 2 };**

***const* source = { b: 4, c: 5 };**

***const* returnedTarget = *Object*.assign(target, source);**

**console.log(target);**

**output: Object { a: 1, b: 4, c: 5 }**

**console.log(returnedTarget);**

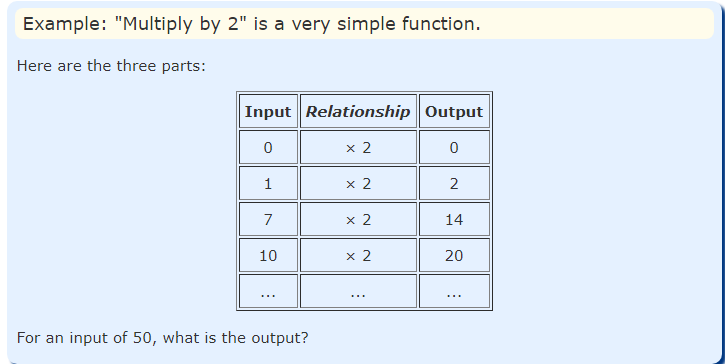
**output: Object { a: 1, b: 4, c: 5 }**

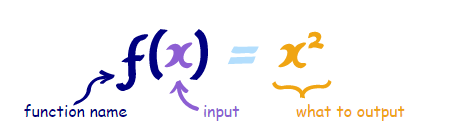
Function

**Input, Relationship/Processing, Output**

We will see many ways to think about functions, but there are always three main parts:

* The input
* The relationship
* The output





Define a Function

function functionName(){

body operation

}

***function* functionName(){**

**console.log("I am a Function")**

**}**

***function* add(){**

***var* a=10;**

***var* b=20;**

**console.log(a+b);**

**}**

***function* sub(){**

***var* a=10;**

***var* b=20;**

**console.log(a-b)**

**}**

\*Invoking a function / Call a Function

**functionName();**

**add();**

**sub()**

**for(*var* i=0;i<=10;i++){**

**functionName()**

**}**

\*\*Function Arguments and Parameters / input

***function* add(*a*,*b*){**

***var* result=*a*+*b*;**

**console.log(result);**

**}**

**add(10,20)**

**add(50,70)**

\*We can also do this

***function* add(*a*,*b*){**

***var* result=*a*+*b*;**

**console.log(result);**

**}**

***var* num1=10;**

***var* num2=20;**

**add(num1,num2)**

\*loop 100 times print

***function* funcName(*i*){**

**console.log(*i*+" Times Print");**

**}**

**for(*var* i=0;i<=100;i++){**

**funcName(i);**

**}**

\*Travers array value using function

***var* arr1=[1,2,3,4,5,6]**

***function* arrCount(*arr*){**

***var* str="";**

**for(*var* i=0;i<*arr*.length;i++){**

**str=`${str}${*arr*[i]}'`**

**}**

**console.log(str)**

**}**

**arrCount(arr1)**

***var* arr1=[1,2,3,4,5,6]**

***var* arr2=[1,62,30,4,50,6]**

***var* arr2=[1,2,83,40,5,6]**

print all this array using one function system

***var* arr1=[1,2,3,4,5,6]**

***var* arr2=[1,62,30,4,50,6]**

***var* arr3=[1,2,83,40,5,6]**

***function* arrCount(*arr*){**

***var* str="";**

**for(*var* i=0;i<*arr*.length;i++){**

**str=`${str}${*arr*[i]}'`**

**}**

**console.log(str)**

**}**

**arrCount(arr1)**

**arrCount(arr2)**

**arrCount(arr3)**

\*\*Add(+) this Array value using one function system

***var* arr1=[1,2,3,4,5,6]**

***var* arr2=[1,62,30,4,50,6]**

***var* arr3=[1,2,83,40,5,6]**

***function* add(*array*){**

***var* sum=0;**

**for(*var* i=0;i<*array*.length;i++){**

**sum+=*array*[i];**

**}**

**console.log(sum);**

**}**

**add(arr1)**

**add(arr2)**

**add(arr3)**

\*\*Argument Object

***function* test(*a*,*b*,*c*){**

**console.log(arguments) //output**

**}**

**test(20,30,40)**

we can also pass argument without parameter

***function* test(){**

**console.log(arguments) //output**

**}**

**test(20,30,40)**

we can access argument like array notation with arguments objects

***function* test(){**

**console.log(arguments['0'])**

**}**

**test(20,30,40)**

print argument value using loop with arguments object

***function* test(a,b,c){**

**for(*var* i=0;i<arguments.length;i++){**

**console.log(arguments[i])**

**}**

**}**

**test(20,30,40)**

add unlimited number

***function* addAll(){**

***var* sum=0;**

**for(*var* i=0;i<arguments.length;i++){**

**sum+=arguments[i];**

**}**

**console.log(sum);**

**}**

**addAll(5,5,5,5,5,5,5,5,5555,8,0,9,8,67,6)**

**addAll(5,5,5,5,5,5,5,5,5555,8,0,9,8,67,6,5,98,65454,564)**

\*\*Function Return

If we do not return function, we can not store argument in a variable like this:

***function* addAll(){**

***var* sum=0;**

**for(*var* i=0;i<arguments.length;i++){**

**sum+=arguments[i];**

**}**

**}**

***var* a=addAll(5,5,5,5,5,5,5,5,5555,8,0,9,8,67,6)**

***var* b=addAll(5,5,5,5,5,5,5,5,5555,8,0,9,8,67,6,5,98,65454,564);**

**console.log(a)//output undefiend**

**console.log(b)//output undefiend**

if we return sum than we can store argument resulted in a variable like this:

***function* addAll(){**

***var* sum=0;**

**for(*var* i=0;i<arguments.length;i++){**

**sum+=arguments[i];**

**}**

**return sum;**

**}**

***var* a=addAll(5,5,5,5,5,5,5,5,5555,8,0,9,8,67,6)**

***var* b=addAll(5,5,5,5,5,5,5,5,5555,8,0,9,8,67,6,5,98,65454,564);**

**console.log(a)//output 5693**

**console.log(b)//output 71814**

if we want return as multiline

***function* person(*name*,*email*){**

***function* person(*name*,*email*){**

**return{   //here "return" returned an**

**objects**

**fullName:*name*,**

**emailAddress:*email***

**}**

**}**

**console.log(person("Rasel","raselhossainy52@gmail.com"))**

if I want print return object single property value

***function* person(*name*,*email*){**

**return{   //here "return" returned an objects**

**fullName:*name*,**

**emailAddress:*email***

**}**

**}**

***var* Name=person("Rasel","sample@gmail.com");**

**console.log(Name.fullName)**

**console.log(Name.emailAddress)**

\*Function Expression

What is a **Function Expression**? A JavaScript **function** can also be defined using an **expression**. A **function expression** can be stored in a variable: var x = **function** (a, b) {return a \* b}; After a **function** expression has been stored in a variable, the variable can be used as a **function**

\*function statement

***function* add(*a*,*b*){**

**return *a*+*b*;**

**}**

**console.log(add(5,6))**

\*Declare Function Expression

***var* addition=*function* add(*a*,*b*){**

**return *a*+*b*;**

**}**

**console.log(addition(5,6))**

Now we have to call function from variable not function name. now variable is function name. if I call function with the name of “Add” It’s return and Error message

***var* addition=*function*(*a*,*b*){ //anonymous function**

**return *a*+*b*;**

**}**

***var* anotherAddition=addition;//Ex func store in another variable**

**console.log(anotherAddition(5,5))**

\*setTimeout function

**setTimeout(*function*(){**

**console.log("hi i am Rasel");**

**},5000)**

\*\*Inner Function / Nested Function

Greeting program :

***function* something(*greet*,*name*){**

***function* sayHi(){**

***var* result= `${*greet*} , ${*name*}`**

**return result;**

**}**

**return sayHi()**

**}**

**console.log(something("Good Morning","Rasel"))**

\*\*Write a program that input “greet and fullname” and output “greet ,first name from fullname” if name is not than return greet

***function* something(*greet*,*name*){**

***function* getFirstName(){**

**if(*name*){**

**return *name*.split(" ")[0];**

**}else{**

**return "";**

**}**

**}**

***var* message=*greet* +" "+getFirstName();**

**console.log(message)**

**}**

***var* a="Good Morning";**

***var* b="Rasel Hossain"**

**// something("Good Modning","Rasel Hossain")**

**something(a,b)**

***function* greeting(*FullName*,*greet*){**

***function* fullName(){**

**if(*FullName*){**

**return *FullName*.split(" ")[0] +" "+ *FullName*.split(" ")[1];**

**}else{**

**return "";**

**}**

**}**

***var* message=`${*greet*} ${fullName()}`;**

**document.write(message);**

**}**

***var* FullName=prompt("Enter Your Name:");**

***var* greet="Good Morning";**

**greeting(FullName,greet)**

\*\*Function Scoping

**var nam="Rasel"; //global scop**

**function name(){**

**console.log(nam);**

**}**

**name();**

**function name(){**

**var nam="Rasel Hossain"; //local scop**

**}**

**name();**

**console.log(nam); *//nam is not defined***

\*\* run a program which is input a number as a argument and check argument number is %5==0 ,%3==0

**function divide(n){**

**function a(){**

**return n%3==0;**

**}**

**function b(){**

**return n%5==0;**

**}**

**if(a() && b()){**

**console.log(`${n} is divisible by both 3 and 5`);**

**}else{**

**console.log('Not a valid number');**

**}**

**}**

**divide(15)**

Functional Programming

Pure Function

* it returns the same result if given the same arguments
* it does not cause any observable side effects

if this all condition is fulfill then we can called it pure function

**this is pure function example:**

**function sqr(n){**

**return n\*n;**

**}**

**console.log( sqr(2) )**

**console.log( sqr(3) )**

**console.log( sqr(5) )**

**var n=10;**

**function change(){**

**n=100;**

**}**

**console.log(n); //outpur =10;**

But if I call change() function than n will be 100 .that’s why It’s not a pure function

**var n=10;**

**function change(){**

**n=100;**

**}**

**change()**

**console.log(n); //output 100**

another un pure function example:

**var point={**

**x:45,**

**y:30**

**}**

**function printPoint(pointPrint){**

**pointPrint.x = 100;**

**pointPrint.b = "New Property",**

**pointPrint.y = 36**

**console.log(pointPrint);**

**}**

**printPoint(point); console.log(point);**

here output will be :

{

X:100,

Y:36,

B: “New Property”

}

So, this is also not a pure function cause the function scope changed the global variable value

First Class Function:

* A function can be stored in a variable
* A function can be stored in an Array (As Element)
* A function can be stored in an object
* We can create function as need

If all this condition is fulfill then we can called it first class function

**function add(a,b){**

**return a+b;**

**}**

**var sum=add;**

***//1. A Function can be stored in a variable***

**var sum=add;**

**console.log(sum(5,5));**

**console.log(typeof(add));**

**console.log(typeof(sum));**

Array

***//2. A function can be stored in an array***

**var arr=[];**

**arr[0]=sum;**

**arr.push(add)**

**console.log(arr);*//here is tow function  0.sum 1.add***

**console.log( arr[0](5,5) );**

Objects

***//3. A function can be stored in an object***

**var obj={**

**sum:add**

**}**

**console.log(obj.sum(4,9));**

***//3. A function can be stored in an object***

**var obj={**

**sum:add,**

**name:sum**

**}**

**console.log(obj.sum(4,9));**

**console.log(obj['name'](10,20));**

***//4. we can create function as need***

**setTimeout(function(){**

**console.log('I have Created ........');**

**},2000)**

Higher Order Function

* We can pass function as an Arguments
* We can return functions from another function

If all this condition is fulfill then we can called it Higher Order function

**function add(a,b){**

**return a+b;**

**}**

**function manipulate(a,b,func){**

**let c=a+b;**

**let d=a-b;**

**function multiply(){**

**var m=func(a,b); *// pass add function as an arguments***

**return c\*d\*m**

**}**

**return multiply();**

**}**

**var multiply=manipulate(3,4,add)//pass a function as argument**

**console.log(multiply);**

**function manupolition(a,b,func){**

**let c=a+b;*//15***

**let d=a-b;*//5***

**return function result(){**

**let m=func(a,b);**

**return c\*d\*m**

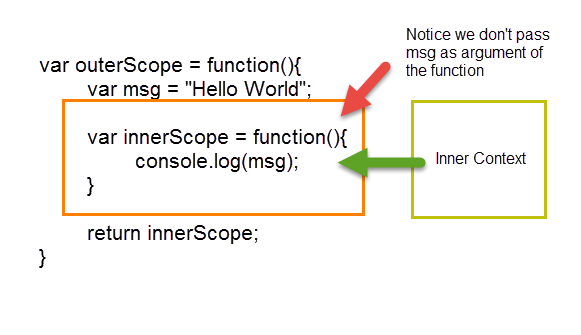
**}**

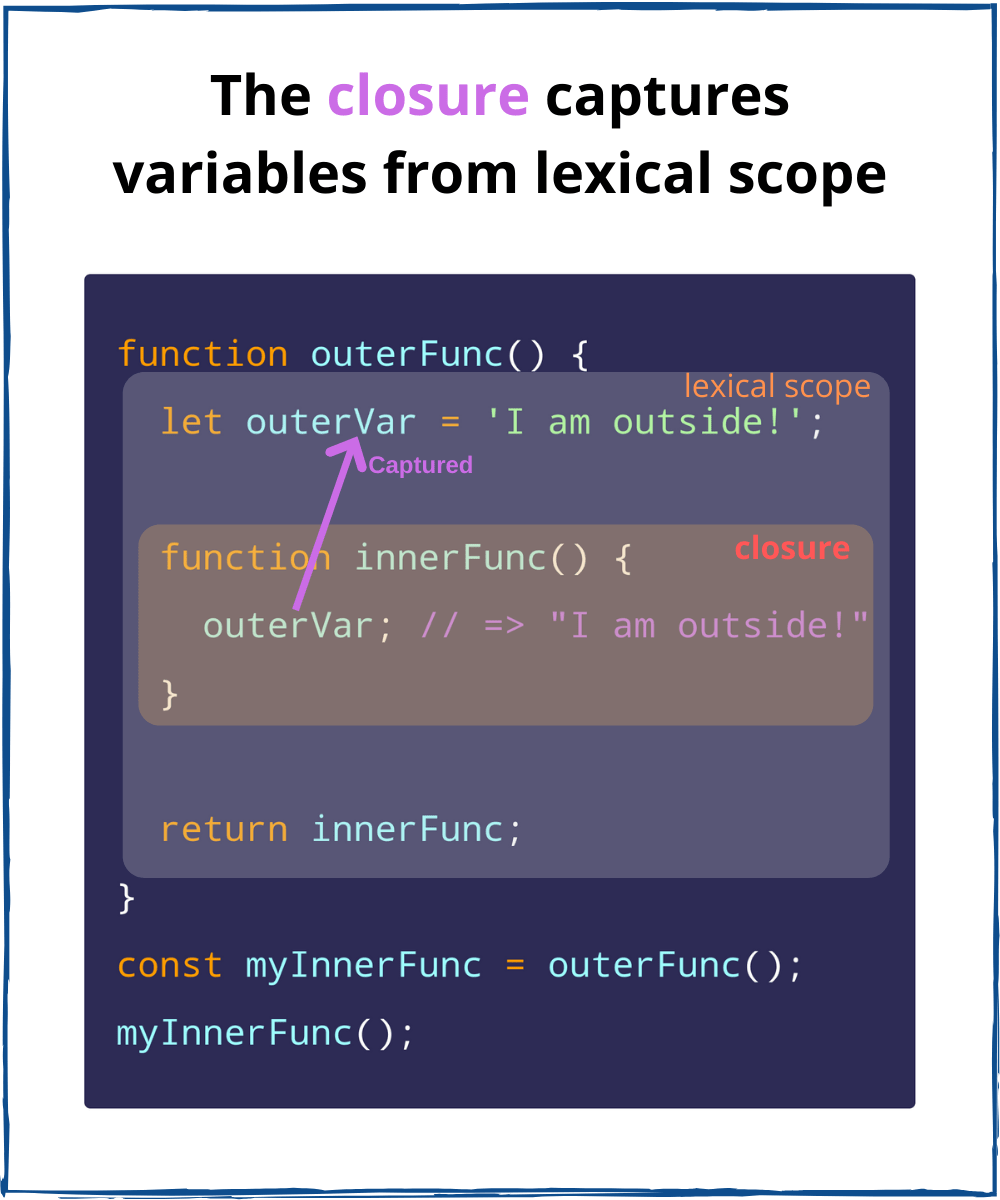
**}**

***// let multiply=;***

**let multiply=manupolition(3,2,add) *// i have to store function in a variable  and input arguments because i return a function in a function***

**console.log(multiply())**

Closure First Look



**let b=10;**

**function a(){**

**console.log(b) *// this is an closure, because var b is global scope and function is another/inner scope.***

**let x=5;**

**return function(){**

**console.log(x) *// this is also a closure, because var x is outer  scope and function is another/inner scope***

**}**

**}**

**let abc=a();**

**console.dir(abc());**

Callback Function

**//This is not callback function**

**function sample(a,b){**

**var c=a+b;*//60***

**var d=a-b;*//40***

**var result=sum(c,d); *//100***

**return result;**

**}**

**function sum(a,b){**

**return a+b; *//100***

**}**

**console.log(sample(50,10))**

**//this is callback function**

**function sample(a,b,callback){**

**let c=a+b;*//30***

**let d=a-b;*//-10***

**let result=callback(c,d);**

**return result;**

**}**

**let result1=sample(10,20,function sample(a,b){**

**return a+b;**

**})**

**let result2=sample(50,60,function(c,d){**

**return c-d;**

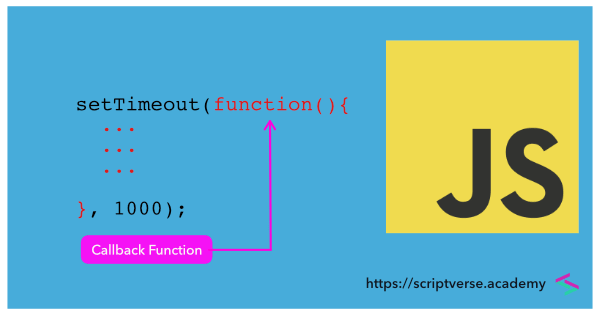
**})**

**let result3=sample(40,70,function(x,y){**

**return x/y;**

**})**

**console.log(result1,result2,result3)**



“Foreach” function

\*travers an array using “forEach” function

This is functional way to travers array

**var arr=[1,2,3,4,5,6,7,8,];**

**arr.forEach(function(val,i,ar){**

**console.log(val)**

**console.log(i)**

**console.log(ar)**

**})**

\*\*array value add

**let arr=[1,2,3,4,5,6,7,8,9];**

**let sum=0;**

**arr.forEach(function(value,index,array){**

**sum+=value;**

**})**

**console.log(sum);**

\*\*travers any array using function

**let arr=[1,2,3,4,5,6,7,8,9];**

**let arr2=[1,2,3,"Rasel"];**

**function forEach(array){**

**for(let i=0;i<array.length;i++){**

**console.log(array[i])**

**}**

**}**

**forEach(arr)**

**forEach(arr2)**

\*\*This is Functional way.and it’s called callback function

**let arr=[1,2,3,4,5,6,7,8,9];**

**let arr2=[1,2,3,"Rasel"];**

**function forEach(array,callback){**

**for(let i=0;i<array.length;i++){**

**callback(array[i],i,array);**

**}**

**}**

**forEach(arr,function(value,index,array){**

**console.log(value)**

**console.log(index)**

**console.log(array)**

**})**

**forEach(arr2,function(value,index,array){**

**console.log(index,value,array)**

**})**

\*\*Sum,sub,multi every array element

**let arr=[1,2,3,4,5,6,7,8,9];**

**function forEach(array,callback){**

**for(let i=0;i<array.length;i++){**

**callback(array[i],i,array);**

**}**

**}**

**let sum=0;**

**var sub=0;**

**let multiple=0;**

**forEach(arr,function(value){**

**sum+=value;**

**sub-=value;**

**multiple\*=value;**

**})**

**console.log(sum)**

**console.log(sub)**

**console.log(multiple)**

**let arr=[1,2,3,4,5,6,7,8,9];**

**function forEach(arr,callback){**

**for(var i=0;i<arr.length;i++){**

**callback(arr[i],i,arr);**

**}**

**}**

**forEach(arr,function(value,index,arr){**

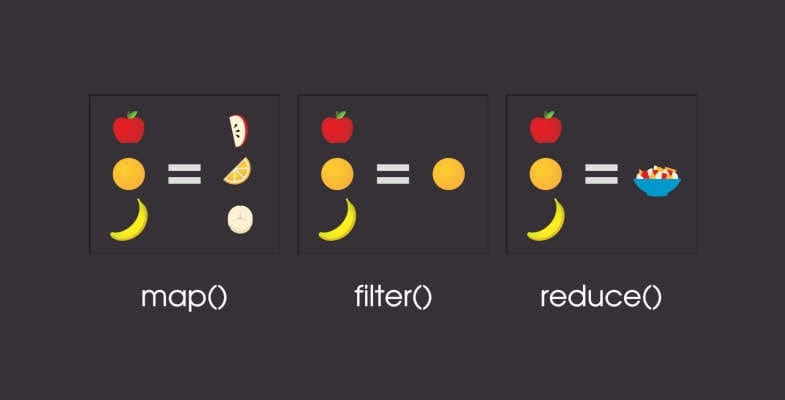
**arr[index] = value+5;**

**})**

**console.log(arr)**

Map Function

* The **map()** method **creates a new array** populated with the results of calling a provided function on every element in the calling array.
* The map() method creates a new array with the results of calling a function for every array element.



\*\*This is changed the main array

**let arr=[1,2,3,4];**

**let result=arr.map(function(value,index,arr){**

**return arr[index]=value+5;**

**})**

**console.log(result)**

**console.log(arr)**

\*map return new array without changing main array like this. But this is only for sqrt not multi purpose

***// traverse array***

***// sqrt array value***

***//creat new\_array***

***// push sqrt in new\_array***

***// return new array***

**let arr=[1,2,3];**

**function map(arr){**

**let newArr=[];**

**for(var i=0;i<arr.length;i++){**

**let sqrt=arr[i]\*arr[i]**

**newArr.push(sqrt);**

**}**

**return newArr;**

**}**

**console.log(map(arr))**

**console.log(arr)**

\*this function can be use for multi purpose

**let arr=[1,2,3];**

**function map(arr,cb){**

**let newArr=[];**

**for(var i=0;i<arr.length;i++){**

**let temp=cb(arr[i],i,arr);**

**newArr.push(temp);**

**}**

**return newArr;**

**}**

**console.log(map(arr,function(value){**

**return value\*\*3;**

**}))**

**console.log(arr)**

**let arr=[1,2,3];**

**function myMap(arr,cb){**

**var newArray=[];**

**for(var i=0;i<arr.length;i++){**

**newArray.push(cb(arr[i],i,arr));**

**}**

**return newArray;**

**}**

**var Pow=myMap(arr,function(value){**

**return Math.pow(value,3);**

**})**

**var mten=myMap(arr,function(value){**

**return value\*10;**

**})**

**console.log(Pow,mten)**

**console.log(arr);**

Filter Function

\*\*Filter function based on logical

\*even or odd number filter function:

**let arr=[1,2,3,4,5,6,7,8,9,10,11]**

**var newArray=arr.filter(function(value){**

**return value%2==0;**

**})**

**console.log(newArray)**

**console.log(arr)**

\*even or odd number filter function:

**var arr=[1,2,3,4,5,6,7,8,9,10,11]**

**function filter(arr,cb){**

**var newArr=[];**

**for(var i=0;i<arr.length;i++){**

**cb(arr[i],newArr);**

**}**

**return newArr;**

**}**

**console.log(filter(arr,function(value,newArr){**

**if(value%2==0){**

**return newArr.push(value);**

**}**

**}))**

**console.log(filter(arr,function(value,newArr){**

**if(value%2==1){**

**return newArr.push(value)**

**}**

**}))**

\*even or odd number filter function another way:

**let arr=[1,2,3,4,5,6,7,8,9,10,11]**

**function filter(arr,cb){**

**let newArray=[];**

**for(let i=0;i<arr.length;i++){**

**cb(arr[i],newArray);**

**}**

**return newArray;**

**}**

***// For Even***

**function even(value,newArray){**

**if(value%2==0){**

**return newArray.push(value);**

**}**

**}**

***// For Odd***

**function odd(value,newArray){**

**if(value%2!=0){**

**return newArray.push(value);**

**}**

**}**

**let EvenNumber=filter(arr,even);**

**let oddNumber=filter(arr,odd)**

**console.log(EvenNumber,oddNumber)**

\*even or odd number filter function(only)change cb in if:

**let arr=[1,2,3,4,5,6,7,8,9,10,11]**

**function filter(arr,cb){**

**let newArray=[]**

**for(var i=0;i<arr.length;i++){**

**if( cb( arr[i] ) ){**

**newArray.push(arr[i]);**

**}**

**}**

**return newArray;**

**}**

***// Even NUmber***

**var evenNumber=filter(arr,function(value){**

**return value%2==0;**

**})**

**console.log(evenNumber);**

***// Odd Number***

**var oddNumber=filter(arr,function(value){**

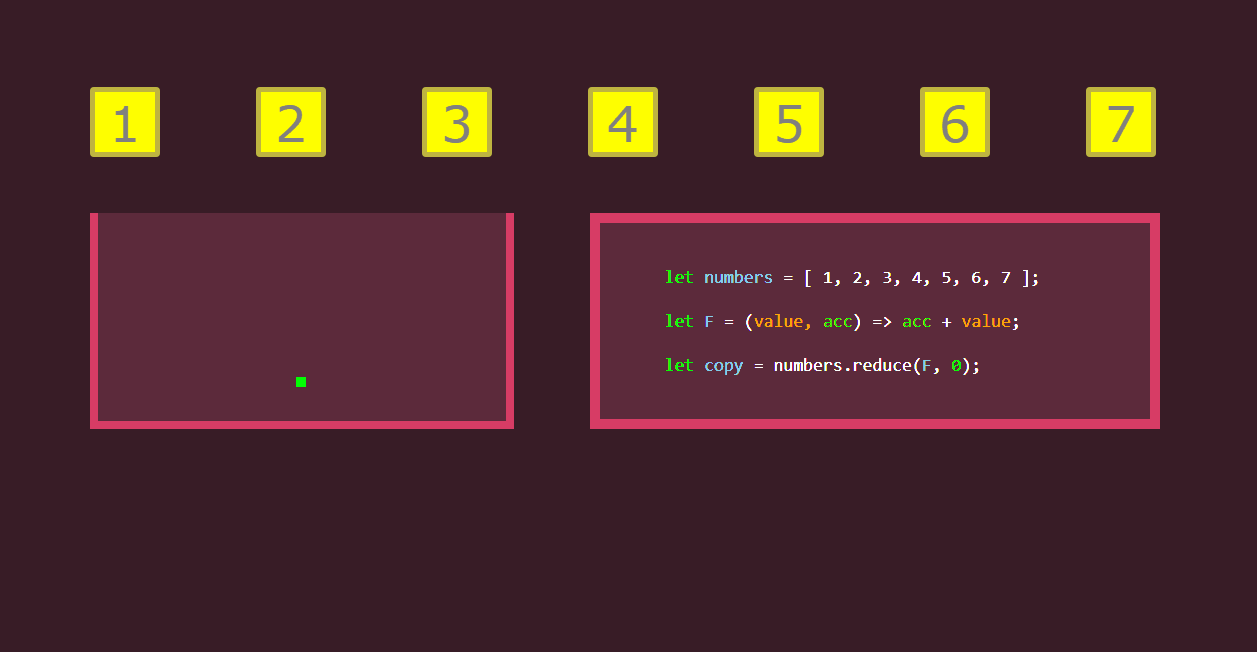
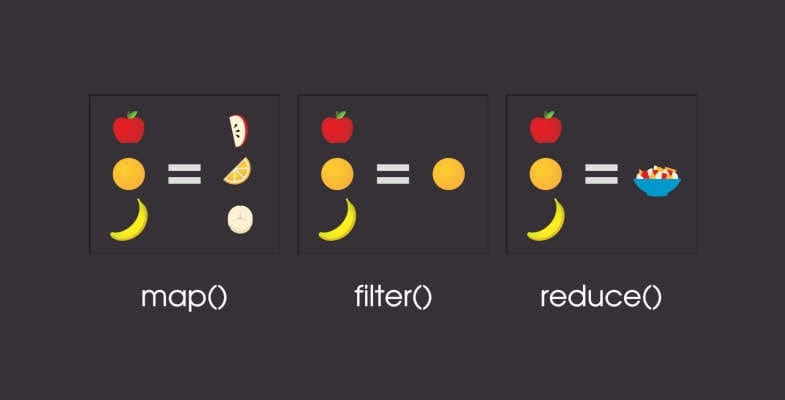
**return value%2!=0;**

**})**

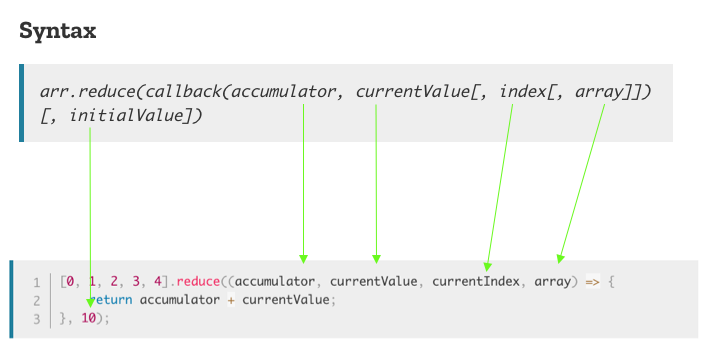
**console.log(oddNumber);**

Reduce Function

The **reduce**() method reduces the array to a single value. The **reduce**() method executes a provided **function** for each value of the array (from left-to-right). The return value of the **function** is stored in an accumulator (result/total). Note: **reduce**() does not execute the **function** for array elements without values.



\*\*Reduce Function Syntax



**var arr=[1,2,3]**

**const printReduce=arr.reduce(function(previous,current,index,array){**

**console.log(previous);**

**console.log(current);**

**console.log(index);**

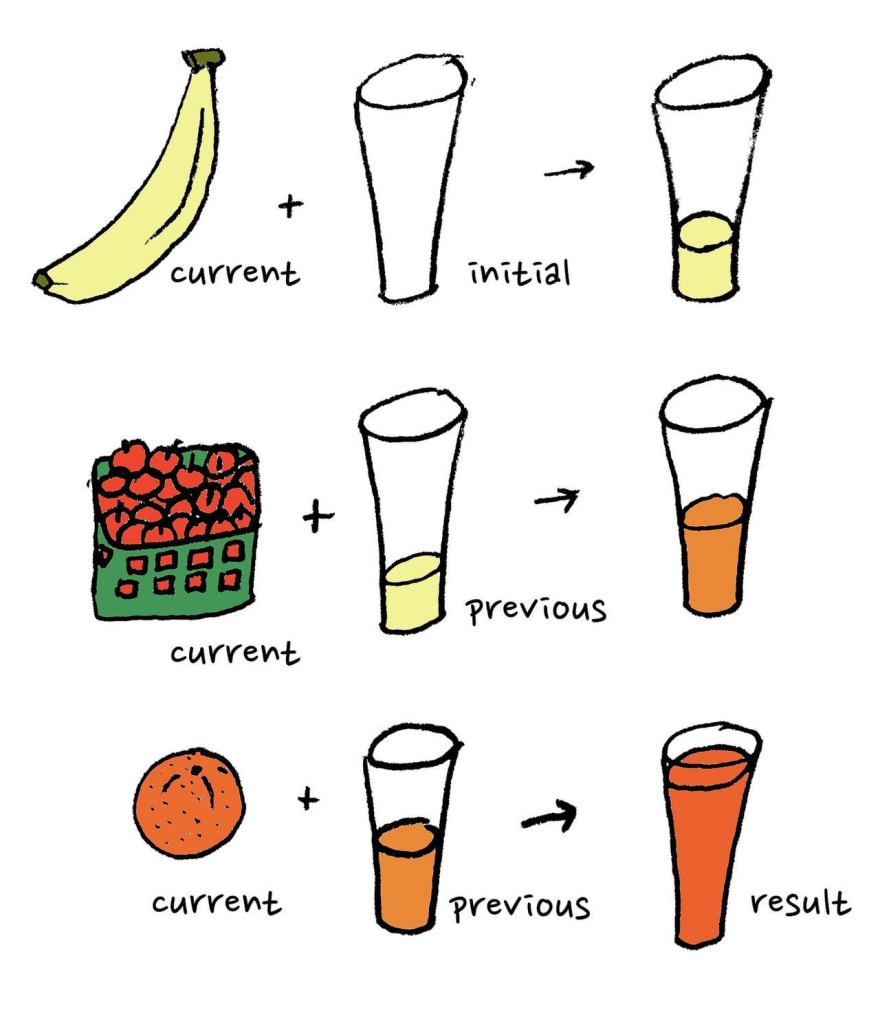
**console.log(array);**

**console.log(array[index]);**

**console.log(array[0]);**

**})**

**console.log(sum);**



\*\*Sum array value with reduce function :

**var arr=[1,2,3]**

**const sum=arr.reduce(function(prev,curr){**

**return prev+curr;**

**})**

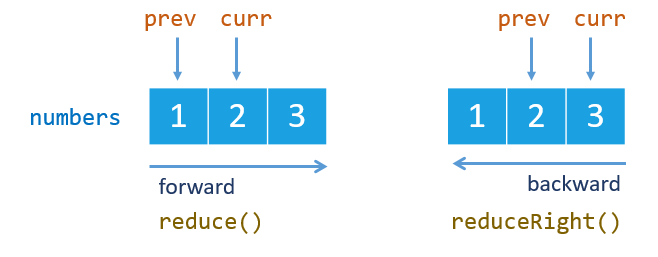
**console.log(sum) //output = 6**

note: reduce function work like this ;

prev + curr = sum

1 + 2 = 3

3 + 3 = 6



\*\*max value

**var arr=[1,2,3,4,5,6,7,8,9,10,11,12]**

**var max=arr.reduce(function(prev,current,value,index,arr){**

**return Math.max(prev+current);**

**})**

**console.log(max)**

\*\*Min

**var arr=[1,2,3,4,5,6,7,8,9,10,11,12]**

**var min=arr.reduce(function(prev,current,value,index,arr){**

**return Math.min(prev+current);**

**},0)**

**console.log(min)**

**var arr=[1,2,3,4,5,6,7,8,9,10,11,12]**

**function myreduce(arr,callback,acc){**

**for(var i=0;i<arr.length;i++){**

**acc=callback(acc,arr[i]);**

**}**

**return acc;**

**}**

***// Sum***

**var sum=myreduce(arr,function(prev,current){**

**return prev+current;**

**},arr[0])**

**console.log(sum)**

***// Max***

**var max=myreduce(arr,function(prev,current){**

**return Math.max(prev,current);**

**},200)**

**console.log(max);**

**var min=myreduce(arr,function(prev,current){**

**return Math.min(prev,current);**

**},-9000)**

**console.log(min)**

Find and FindIndex

Find value

**var arr=[1,2,3,4,5,6,7,8,9]**

**var result=arr.find(function(value){**

**return value==9;**

**})**

**console.log(result);**

find index:

**var arr=[1,2,3,4,5,6,7,8,9]**

**var result2=arr.findIndex(function(value){**

**return value===9;**

**})**

**console.log(result2);**

find value with callback function

**console.log(result)**

**function myfind(arr,cb){**

**for(var i=0;i<arr.length;i++){**

**if(cb(arr[i])){**

**return arr[i];**

**}**

**}**

**}**

**var result2=myfind(arr,function(vlaue){**

**return vlaue==10;**

**})**

**console.log(result2)**

find index from value with callback function:

**function myfind(arr,cb){**

**for(var i=0;i<arr.length;i++){**

**if(cb(arr[i])){**

**return i;**

**}**

**}**

**}**

**var result2=myfind(arr,function(vlaue){**

**return vlaue==9;**

**})**

**console.log(result2)**

Sort, Some, Every

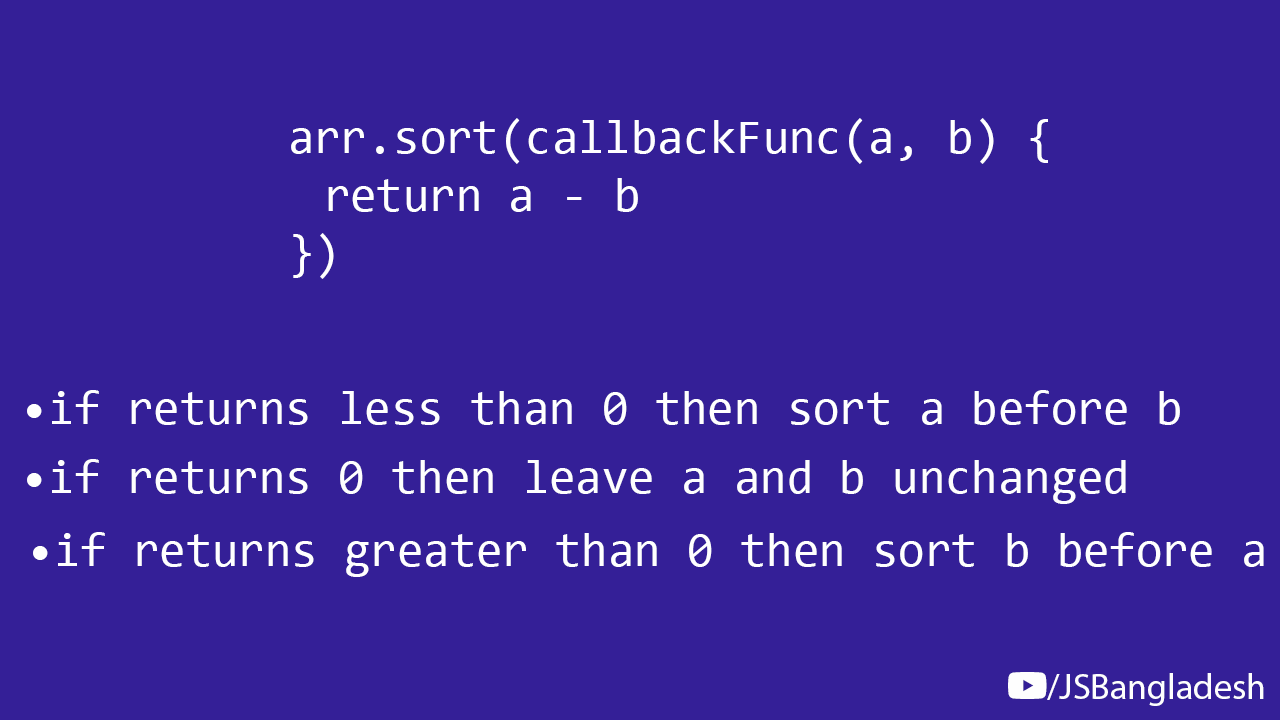
Sort :

Note: this sort method give wrong answer

**var arr=[1,2,3,4,5,6,7,8,9,10,11];**

**arr.sort()**

**console.log(arr);**



Right way: small to big number

**var arr=[1,2,3,4,5,6,7,8,9,10,11];**

**arr.sort(function(a,b){**

**return a-b;**

**})**

**console.log(arr)**

small to big number

**var arr=[1,2,3,4,5,6,7,8,9,10,11];**

**arr.sort(function(a,b){**

**if(a>b){**

**return 1;**

**}else if(a<b){**

**return -1;**

**}else{**

**return 0;**

**}**

**})**

**console.log(arr)**

Big to Small number:

**var arr=[1,2,3,4,5,6,7,8,9,10,11];**

**arr.sort(function(a,b){**

**return b-a;**

**})**

**console.log(arr)**

Big to Small number:

**var arr=[1,2,3,4,5,6,7,8,9,10,11];**

**arr.sort(function(a,b){**

**if(a>b){**

**return -1;**

**}else if(a<b){**

**return 1;**

**}else{**

**return 0;**

**}**

**})**

**console.log(arr)**

\*\*Object Sort (small to big)

**var person=[**

**{**

**name:'A',**

**age:24**

**},**

**{**

**name:'B',**

**age:19**

**},**

**{**

**name:'C',**

**age:26**

**},**

**{**

**name:'D',**

**age:21**

**}**

**]**

**person.sort(function(a,b){**

**return a.age-b.age;**

**})**

**console.log(person)**

another way to sort object (small to big)

**person.sort(function(a,b){**

**if(a.age>b.age){**

**return 1;**

**}else if(a.age<b.age){**

**return -1;**

**}else{**

**return 0;**

**}**

**})**

**console.log(person)**

**var arr=[1,2,3,4,5,6,7,8,9,10,11];**

**var result=arr.every(function(value){**

**return value>=100;**

**})**

**console.log(result)**

\*\*some

The **some()** method tests whether at least one element in the array passes the test implemented by the provided function. It returns a Boolean value.

**var arr=[1,2,3,4,5,6,7,8,9,10,11];**

**var result=arr.some(function(value){**

**return value%2==0;**

**})**

**console.log(result)**

**var arr=[1,2,3,4,5,6,7,8,9,10,11];**

**var result=arr.some(function(value){**

**return value>=100;**

**})**

**console.log(result)**

\*\*Every

The **every()** method tests whether all elements in the array pass the test implemented by the provided function. It returns a Boolean value.

**var arr=[1,2,3,4,5,6,7,8,9,10,11];**

**var result=arr.every(function(value){**

**return value%2==0;**

**})**

**console.log(result)**

\*\*Return a Function from another function

**function greet(mags){**

**function greeting(name){**

**return mags+" , "+name+"!";**

**}**

**return greeting;**

**}**

**var gm=greet("Good Morning");**

**var msg=gm("Rasel");**

**var gn=greet("Good Night");**

**var msg2=gn("Rasel Hossain Niloy")**

**console.log(msg,msg2)**

**console.log(gm("Sakib")) //Good Morning , Sakib**

\*\*pow

**function base(b){**

**return function(n){**

**var result=1;**

**for(var i=0;i<b;i++){**

**result\*=n;**

**}**

**return result;**

**}**

**}**

**var base10=base(10);**

**console.log(base10(2));**

Recursive Function

\*print hellow world 10 times without loop

**//1 to 10**

**function sayHi(n){**

**if(n==10){**

**return**

**}**

**console.log(n+" Hello World");**

**sayHi(n+1)**

**}**

**sayHi(1)**

10 to 1

**function sayHi(n){**

**if(n==0){**

**return**

**}**

**console.log(n+" Hello World");**

**sayHi(n-1)**

**}**

**sayHi(10)**

\*sum

**function sum(n){**

**if(n==1){**

**return 1;**

**}**

**return n+sum(n-1);**

**}**

**console.log(sum(2))**

\*\* Factorial

**function fact(n){**

**if(n===1){**

**return 1;**

**}**

**return n\*fact(n-1);**

**}**

**var reuslt=fact(5);**

**console.log(reuslt)**

Currying

**function currying(a){**

**return function(b){**

**return function(c){**

**return a+b+c;**

**}**

**}**

**}**

**var result=currying(5)(10)(15)**

**console.log(result)**

Function Composition

Function composition is the pointwise application of one function to the result of another

**function print(inpu){**

**console.log(inpu)**

**}**

**function mltiplyByFive(n){**

**return n\*5;**

**}**

**function add(a,b){**

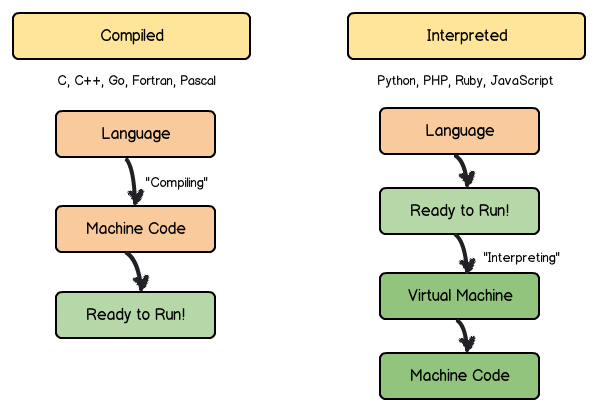
**return a+b;**

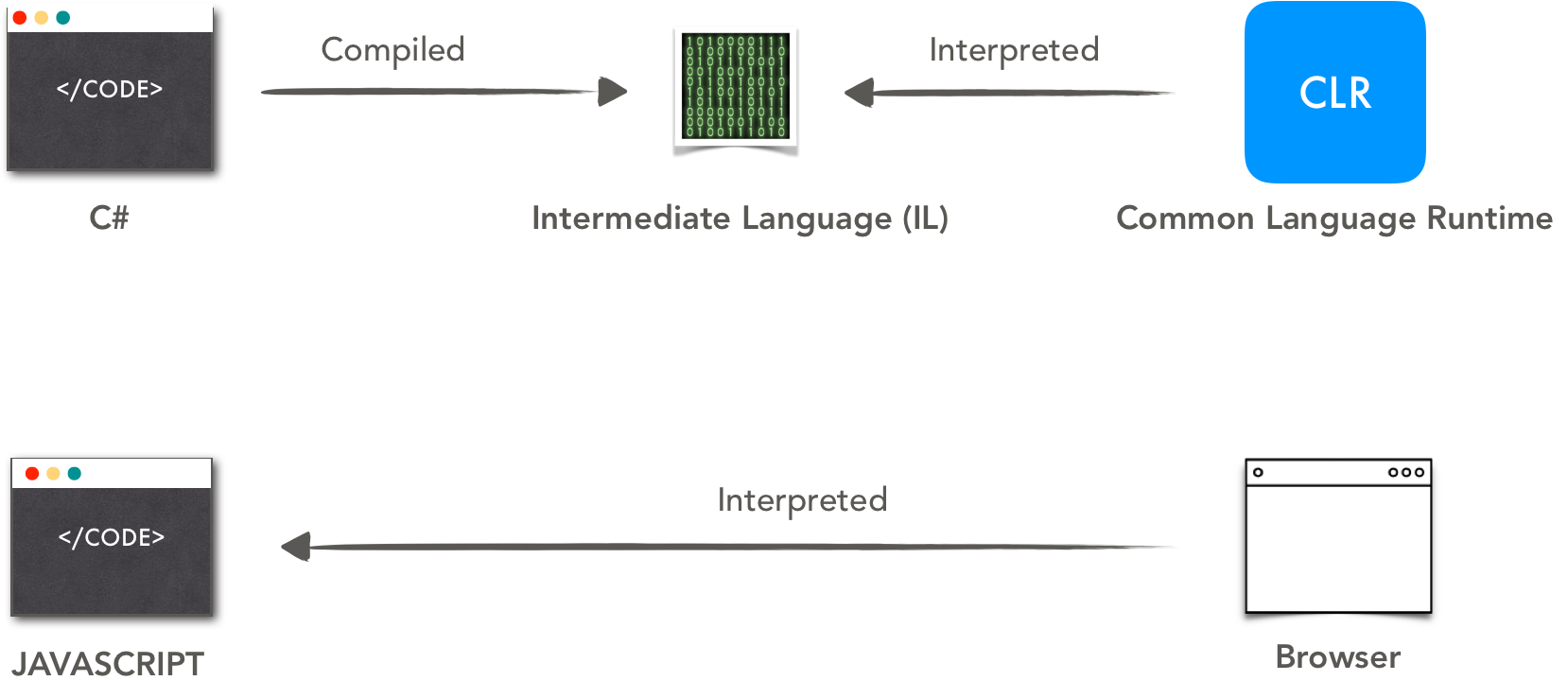
**}**

**print(mltiplyByFive(add(3,5)))**

Compiled vs interpreted

**JavaScript** is an **interpreted** language, not a **compiled** language. A program such as C++ or Java needs to be **compiled** before it is run. ... More modern browsers use a technology known as Just-In-Time (JIT) **compilation**, which **compiles JavaScript** to executable bytecode just as it is about to run.





**Execution Context**

**function a(){**

**b()**

**console.log("I am Function A");**

**}**

**function b(){**

**d()**

**console.log("I am Function b")**

**}**

**function c(){**

**console.log("I am Function C")**

**}**

**function d(){**

**c()**

**console.log("I am Function d")**

**}**

**var x=100;**

**a()**

**console.log("I am Global")**

Creation and Execution of context

1. Creational Phase

2. Executional Phase

**function a(){**

**d()**

**console.log("I am Function A");**

**}**

**function c(){**

**console.log("I am Function C")**

**}**

**function d(){**

**c()**

**console.log("I am Function d")**

**}**

**var x=100;**

**a()**

**console.log("I am Global")**

**//variable context**

**var a=10;**

**var b;**

**console.log(a)**

**console.log(b)**

**b=20;**

**console.log(c)**

**var c=50;**

**console.log(c);**

***/\****

***Creational phase:***

***a = undefined;***

***b = undefined;***

***c = undefined;***

***Executional Phase:***

***a = 10;***

***console.log(a) = 10;***

***console.log(b) = undefined;***

***b = 20;***

***console.log(c) = undefined;***

***c = 50;***

***c***

**//Function Context**

**abc()**

**function abc(){**

**console.log("i am function");**

**}**

**abc()**

***/\****

***Creational phase:***

***function abc = reference /not undefined***

***Executional phase:***

***abc() call***

***abc() call***

***\*/***

**Hoisting**

**var a=100;**

**newPrint(a) *// newPrint is not Defined***

**print(10)**

**var newPrint=print;**

**newPrint(45)**

**function print(a){**

**console.log(a)**

**}**

**print(a)**

***// Creational phase***

***// a= undefined;***

***// newPrint=undefined;***

***// function print() = referance;***

***// Executional Phase***

***// a=100***

***// newPrint(a) call and return Undefined/error***

***// print (10) call print 10;***

***// newPrint=print()***

***// newPrint() call print 45***

***// print(a) call print 100***

Function Declaration vs Expression

**abc()**

**function abc(){**

**console.log("I am Function")**

**}**

**newAbc();**

**var newAbc=function(){**

**console.log("I am New ABC Function")**

**}**

**newAbc()**

***// Creational phase***

***// abc = reference***

***// newAbc()= undefined;***

***//Executional phase***

***// abc() call print "I am Function";***

***// newAbc() return newAbc is not a function / print error msg***

***//newAbc() call print "I am New ABC Function"***

Scope

**var a=11;**

**function A(){**

**var b=12;**

**function B(){**

**var c=23;**

**console.log(c)**

**}**

**function C(){**

**var d=56;**

**console.log(d);**

**}**

**console.log(b)**

**D(b)**

**B()**

**C()**

**}**

**function D(n){**

**return n+a;**

**}**

Closure

**function test(){**

**var msg="I am Learning Lexical Scope and Closure";**

**return function(){**

**console.log(msg)**

**}**

**}**

**var result=test();**

**result()**

**function test(){**

**var msg="I am Learning Lexical Scope and Closure";**

**return function(){**

**return 5+5;**

**}**

**}**

**var result=test();**

**console.log(result());**

closure and loops

**for (var i=1;i<=5;i++){**

**(function(n){**

**setTimeout(function(){**

**console.log(n)**

**},1000\*i)**

**})**

**(i)**

***//Another way***

**function test(n){**

**setTimeout(function(){**

**console.log(n)**

**},1000\*n)**

**}**

**test(i);**

**}**

**OOP**

Normal way:

**var width=10;**

**var height=20;**

**function calculateArea(width,height){**

**return width\*height;**

**}**

**function calculateRange(width,height){**

**return 2\*(width+height);**

**}**

**var area=calculateArea(width,height);**

**var range=calculateRange(width,height);**

**console.log(area,range)**

OOP way:

**var rec={**

**width:10,**

**height:20,**

**calculateArea:function(){**

**return rec.width\*rec.height;;**

**},**

**calculateRange:function(){**

**return 2\*(rec.width+rec.height)**

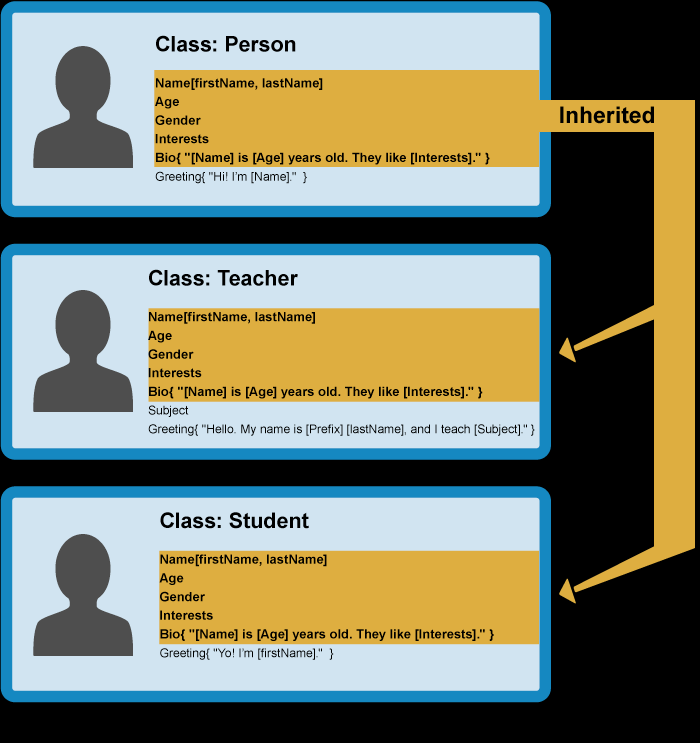
**}**

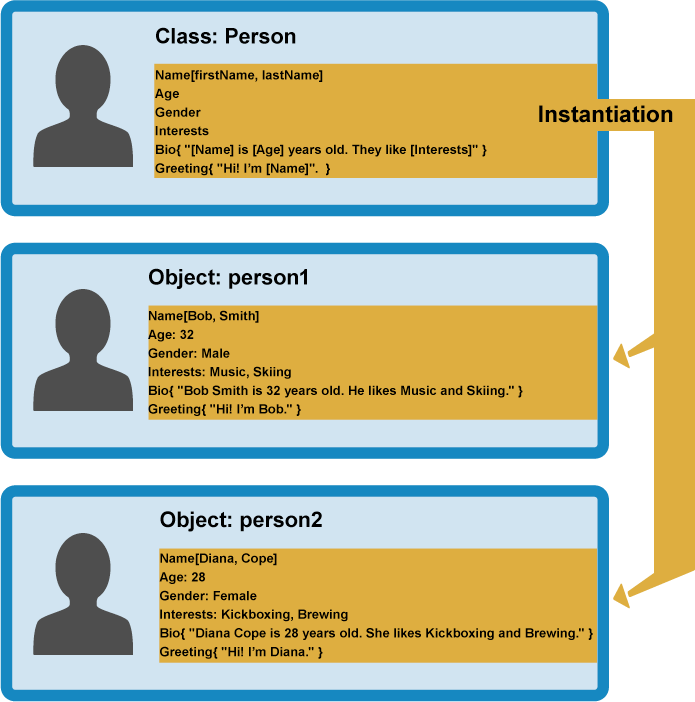
**}**

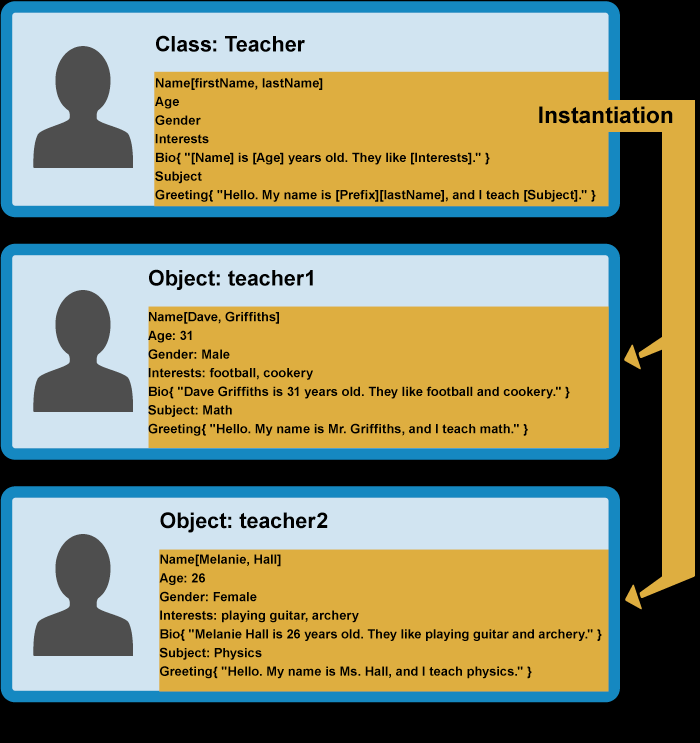
**var area=rec.calculateArea();**

**var range=rec.calculateRange();**

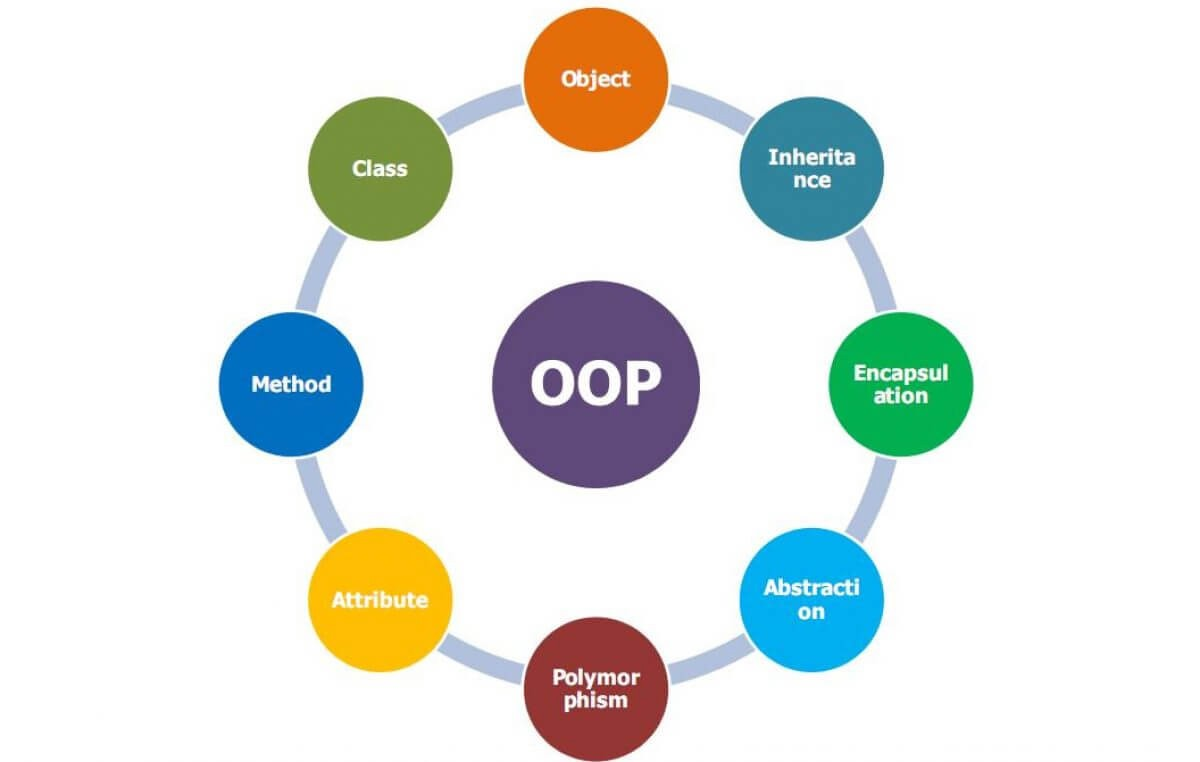
**console.log(area,range)**







Oop Pillars:



Object Literal

**var rect={**

**width:100,**

**height:50,**

**draw:function (){**

**console.log("I am a Rectangle");**

**console.log("My width is "+*this*.width);**

**console.log(`My Height is ${*this*.height}`)**

**}**

**}**

**rect.draw();**

**rect.height=900;**

**rect.draw();**

this

**var rect={**

**width:100,**

**height:50,**

**draw:function(){**

**console.log("I am Rectangle");**

***// call printProperties method with this***

***this*.printProperties();**

**console.log(*this*)**

**},**

**printProperties:function(){**

**console.log("My Width is "+*this*.width);**

**console.log("My Height is "+*this*.height);**

**}**

**}**

**rect.draw(); //outpur: width 100,height 50**

**var another={**

**width:42,**

**height:56,**

**print:rect.printProperties,**

**}**

**another.print() //output: width 42,height:56**

\*\*Create rectangle object 4 times : (note ) :

This is normal way , this is not for multi purpose

**let rect1={**

**width:100,**

**height:50,**

**draw:function (){**

**console.log("I am a Rectangle ")**

***this*.printProperties();**

**},**

**printProperties:function (){**

**console.log(`my width is ${*this*.width}`)**

**console.log(`My height is ${*this*.height}`)**

**}**

**}**

**let rect2={**

**width:100,**

**height:50,**

**draw:function (){**

**console.log("I am a Rectangle ")**

***this*.printProperties();**

**},**

**printProperties:function (){**

**console.log(`my width is ${*this*.width}`)**

**console.log(`My height is ${*this*.height}`)**

**}**

**}**

**let rect3={**

**width:100,**

**height:50,**

**draw:function (){**

**console.log("I am a Rectangle ")**

***this*.printProperties();**

**},**

**printProperties:function (){**

**console.log(`my width is ${*this*.width}`)**

**console.log(`My height is ${*this*.height}`)**

**}**

**}**

This all way does not fulfill OOP rule :

The rule is “ do not repeat yourself ”

In this situation we can use Factory Pattern

Factory Pattern

**var createRect=function(width,height){**

**return {**

**width:width,**

**height:height,**

**draw:function(){**

**console.log("I am a Rectangle");**

**},**

**printProperties:function(){**

**console.log("My width is "+*this*. width);**

**console.log("my height is "+*this*.height);**

**}**

**}**

**}**

**var rect1=createRect(10,8)**

**rect1.draw();**

**var rect2=createRect(40,30)**

**rect2.draw()**

**\*\*Another way**

**function myRectangle(width,height){**

**var rect={**

**width:width,**

**height:height,**

**draw:function(){**

**console.log(`i am rectangle`);**

***this*.printProperties();**

**},**

**printProperties:function(){**

**console.log(`My width is ${*this*.width}`)**

**console.log(`My height is ${*this*.height}`)**

**}**

**}**

**return rect;**

**}**

**var rect1=myRectangle(50,60);**

**rect1.draw()**

Constructor Pattern

Note: If I use constructor pattern than I have to use “this.” Before every property

**\*\*this function return an Error. because "this” keyword should be use in an Object**

**var Rectangle=function(){**

***this*.width=width**

***this*.height=height**

***this*.draw=function(){**

**console.log(`i am rectangle`);**

***this*.printProperties();**

**}**

***this*.printProperties=function(){**

**console.log(`My width is ${*this*.width}`);**

**console.log(`My height is ${*this*.height}`);**

**}**

**}**

**var rect2=Rectangle(10,80); rect2.draw();**

so ,the solution is =>

var rect1=new functionName(arguments)

rect1.propertyName/Method()

**var Rectangle=function(width,height){**

***this*.width=width**

***this*.height=height**

***this*.draw=function(){**

**console.log(`i am rectangle`);**

***this*.printProperties();**

**}**

***this*.printProperties=function(){**

**console.log(`My width is ${*this*.width}`);**

**console.log(`My height is ${*this*.height}`);**

**}**

**}**

**var rect=new Rectangle(50,60);**

**rect.draw()**

New keyword

The **new keyword** is used in JavaScript to create a object from a constructor function.

“Constructor Property” not “Constructor pattern”

**var obj={}; */\*Means :-\*/* var obj=new Object();**

**var str=new String("Rasel")**

**console.log(str)**

**console.log(str.constructor)**

**console.log("My name is "+str) *// output My name is Rasel***

Prove Function is a Object

**var rect=new Function('width','height',**

**`this.width=width;**

**this.height=height;**

**this.draw=function(){**

**console.log("I am Rectangle");**

**this.printProperties();**

**};**

**this.printProperties=function(){**

**console.log("My width is "+ this.width)**

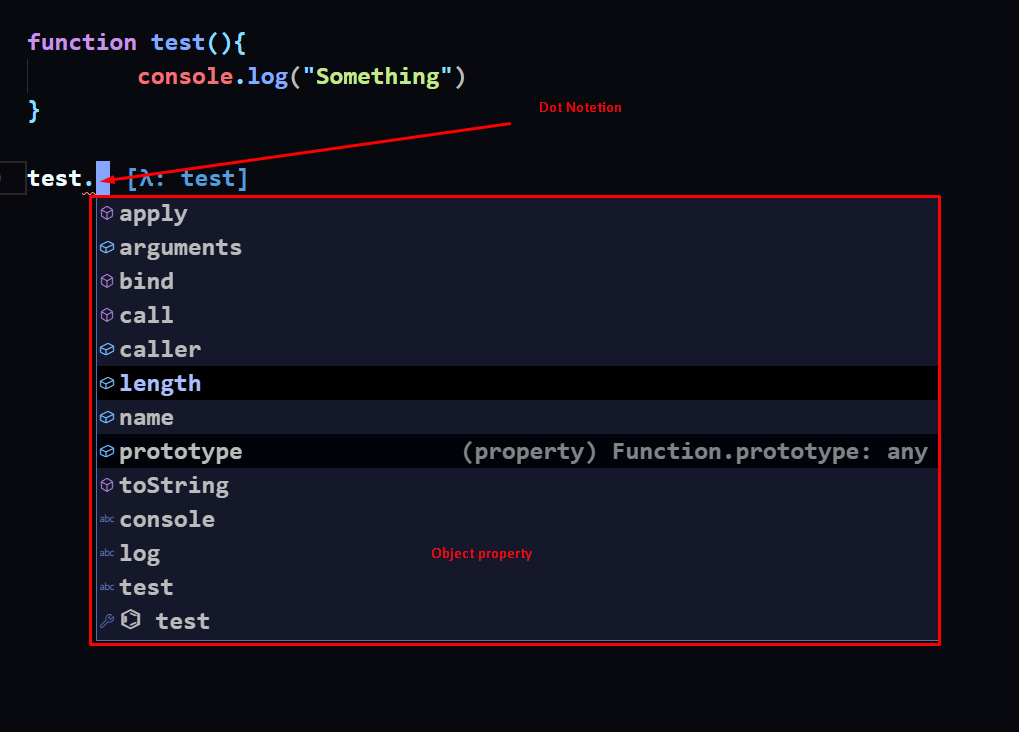
**console.log("My height is "+this.height)**

**}`**

**)**

**var rect5=new rect(4,5)**

**rect5.draw()**



That’s why this is object

Bind Call Apply

**console.log(Rectangle.name) *//function name***

**console.log(Rectangle.length) *//function arguments***

\*Call

**function myFunc(a,b){**

**console.log(*this*.a + *this*.b)**

**}**

**myFunc.call({a:2, b:2})**

**myFunc.call({a: 400,b: 250})**

\*Example 1 of .call:

**var person = {**

**fullName: function() {**

**return *this*.firstName + " " + *this*.lastName;**

**}**

**}**

**var person1 = {**

**firstName:"John",**

**lastName: "Doe"**

**}**

**person.fullName.call(person1);**

\*Example Two of .call:

**var person={**

**fullName:function(){**

**return *this*.firstName + " " + *this*.lastName;**

**}**

**}**

**var persone1={**

**firstName:"Rasel",**

**lastName:"Hossain"**

**}**

**var result=person.fullName.call(persone1)**

**console.log(result)**

Example 3 of .call :

**function studentName(){**

**return *this*.firstName + " "+ *this*.lastName;**

**}**

**var result=studentName.call({firstName:"Rasel",lastName:"Hossain"})**

**console.log(result)**

Example 4 of .call :

**function studentName(){**

**return *this*.firstName + " "+ *this*.lastName;**

**}**

**var student1={**

**firstName:"Rasel",**

**lastName:"Hossain"**

**}**

**var result=studentName.call(student1);**

**console.log(result)**

\*call with arguments Example 5:

**function studentName(country,city){**

**console.log("Country: "+country);**

**console.log("City: "+city);**

**return *this*.a + *this*.b;**

**}**

**var result=studentName.call( {a:20,b:30} , "Bangladesh" , "Chandpur" );**

**console.log(result)**

\*call with arguments Example 6:

**function studentName(country,city){**

**function stData(){**

**return country + " " + city;**

**}**

**return *this*.firstName + *this*.lastName +" "+ stData();**

**}**

**var students2={**

**firstName:"Rasel ",**

**lastName:"Hossain"**

**}**

**var result=studentName.call(students2,"Bangladesh","Chandpure")**

**console.log(result)**

\*\*Apply

Call method and apply both are same . but “call” can receive arguments directly but “apply” can not receive arguments directly.

“Apply” receive arguments as an Array

Example 1 of apply:

**var sum=function(a,b){**

**return *this*.sumText + " is "+ (a+b);**

**}**

**var totalText={**

**sumText:"Total"**

**}**

**var result=sum.apply(totalText, [20, 30]);**

**console.log(result);**

Example 2 of apply:

**function stData(country,city){**

**function details(){**

**return country+city;**

**}**

**return *this*.firstName + " " + *this*.lastName + details();;**

**}**

**var student1={**

**firstName:"Rasel",**

**lastName:"Hossain "**

**}**

**var result=stData.apply(student1,["Bangladesh ","Chandpur "])**

**console.log(result)**

\*\*Bind

Bind Method dose not call instantly like “call” and “apply” method

Like this:

**function myFunc(c,d){**

**console.log( (*this*.a + *this*.b) + (c+d) )**

**}**

**myFunc.bind({a:10,b:20},50,60) //nothing output**

**myFunc.call({a:10,b:20},50,60) //output:140**

**myFunc.apply({a:10,b:20},[50,60]) //outpur:140**

this bind method can not call myFunc

we can pass argument when we call bind method function like this:

**function myFunc(c,d){**

**console.log( (*this*.a + *this*.b) + (c+d) )**

**}**

**var result=myFunc.bind({a:10,b:20});**

**result(5,5)**

note: better, pass arguments when call bind method function

\*\*Pass by Value

Vs  
\*\*Pass by Reference

Abstraction: (Private Properties)

**var Rectangle=function(width,height){**

***this*.width=width;**

***this*.height=height;**

**var printProterties=function(){**

**console.log("My height is "+*this*.height);**

**console.log("My width is "+*this*.width)**

**}.bind(*this*);**

***this*.draw=function(){**

**console.log("I am Rectangle");**

**printProterties()**

**}**

**}**

**var rect1=new Rectangle(50,60)**

**rect1.draw()**

\*Getter and Setter private property

There is two way to getter:

1 is:

**var Rectangle=function(width,height){**

***this*.width=width;**

***this*.height=height;**

**var position={**

**x:50,**

**y:190**

**}**

***this*.position=function(){**

**return position;**

**}**

**}**

**var rect1=new Rectangle(50,60)**

**var result=rect1.position()**

**console.log(result)**

Object.defineProperty(object, property, descriptor)

2 is:

**var Rectangle=function(width,height){**

***this*.width=width;**

***this*.height=height;**

**var position={**

**x:50,**

**y:190**

**}**

**Object.defineProperty(*this*,"position",{**

**get:function(){**

**return position;**

**}**

**})**

**}**

**var rect1=new Rectangle(10,50);**

**var result=rect1.position;**

**console.log(result.x);**

Access function scope property:

**function someNumber()**

**{**

**var width = 20;**

**var height = 30;**

***this*.megerment=function(){**

**return width + " "+ height;**

**}**

**}**

**var result=new someNumber();**

**var result2=result.megerment()**

**console.log(result2)**

**function someNumber()**

**{**

**var width = 20;**

**var height = 30;**

**Object.defineProperty(*this*,"meger",{**

**get:function(){**

**return width + " " + height;**

**}**

**})**

**}**

**var result=new someNumber();**

**console.log(result.meger);**