var result=1;

var itrresult=1;

function factorial(n){

result=result

\*n;

if(n==1||n==0)

return result

else

return factorial(n-1)

}

function iterativefactorial(n){

//result=result\*n;

var factorial=1;

if(n==1||n==0){

return n;}

else{

for(var i=1;i<=n;i++){

factorial=factorial\*i;

}

}

return factorial;

}

console.log("Hello world!"+factorial(3));

console.log("Hello world! Iterative factorial"+iterativefactorial(4));

//alert("Hi programer");

// object

var car={

color:"metali-gray",

model:"xe",

price:20000

};

console.log(car.color);

//arrays and objects

var car={

color:"metali-gray",

model:"xe",

price:20000,

type:null

};

car.type='toyota';

car.color='new';

car.contents={

name:"honda",

value:"very much"

}

console.log(car.color+"\t"+car.type);

console.log(car["type"]);

console.log(car.contents.name);

var array=[1,2,3,4];

console.log(array[0]+"\t"+array.length);

var emptyarray=[];

console.log(emptyarray.length);

array[5]=6;

console.log(array[9]);//No array out of bounds exception and no static length array

emptyarray[5]="two hundread";

console.log(emptyarray[0]+"\t"+ emptyarray[5]);

//oject with arrays

var marks={

Daa:[10,29,28,32],

Gt:[19,20,74,94]

}

console.log(marks.Daa[3]);

//arrays with objects

var arraywithobj=[marks={

college:"UNO"

},car];

console.log(arraywithobj[0].college);

//multidimenional array

var data={

trail1:[[1,2],[3,4]]

}

console.log(data.trail1[1][1]);

console.log(1/0);

// functions & functions with objects—methods

//No overloading, no need to retuntype and var in formal parameter

var gvar;

function x(value){

val=value+1;

console.log(val);

return val;

}

function x(value){

val=value-1;

return val;

}

console.log("\t"+x(3));

function mul(v1,v2){

this.gvar=v1+v2;

return v1\*v2;

}

var multiply= mul(3,4);

console.log("\t"+multiply);

var array1=[];

array1.push(1,3,5,6);

array1.reverse();

console.log(array1);

var obj={};

var obj1={};

obj =new mul(1,2);

obj1.multiplication=mul(2,4);

var obj2=new mul(3,3);

console.log("method"+obj.mutiplication+"\t"+obj1.multiplication+"\t"+obj2.gvar+"\t"+obj.gvar);