Deepak K

717821e113

//task1 Recursion and stack:

function recursion(num1){

    if(num1==0){

    return 1;

    }

    return num1\*recursion(num1-1);

}

let num=prompt("enter the number");

let fact=recursion(Number(num));

document.querySelector("h1").innerHTML=fact;

//task2

function fibo(n){

    if(n==1){

        return 0;

    }else if(n==2){

        return 1;

    }else{

    return fibo(n-1)+fibo(n-2);

    }

}

let n1=prompt("enter number to find the nth fibo");

let fib=fibo(Number(n1));

document.querySelector("h1").innerHTML=fib;

//task3

function noOfWays(n){

    if(n==0){

        return 1;

    }else if(n<0){

        return 0;

    }else{

        return noOfWays(n-3)+noOfWays(n-2)+noOfWays(n-1);

    }

}

let n=prompt("enter number steps:");

document.write(noOfWays(n));

//task4

const arr = [[1,2],[3,[4,[5]]]];

const flattened = arr.flat(Infinity);

console.log(flattened);

//task5

function towerOfHanoi(n, from\_rod,  to\_rod,  aux\_rod){

        if (n == 0)

        {

            return;

        }

        towerOfHanoi(n - 1, from\_rod, aux\_rod, to\_rod);

        document.write("Move disk " + n + " from rod " + from\_rod +

        " to rod " + to\_rod+"<br/>");

        towerOfHanoi(n - 1, aux\_rod, to\_rod, from\_rod);

    }

    var N = 3;

    towerOfHanoi(N, 'A', 'C', 'B');

//Task 1 JSON and variable length arguments/spread syntax:

function add(){

    var sum=0;

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

      sum+=arguments[i];

    }

    return sum;

}

console.log(add(1,2,3,4,5));

console.log(add(1,2,3,4,5,6));

console.log(add(1,2,3,4,5,6,7));

//Task 2

function sum(...args){

  let s=0;

  for(let arg of args){

    s+=arg;

  }

  return s;

}

console.log(sum(1,2,3));

// Task 3

var s1={

  name:"danush",

  id:123

}

var s2=JSON.stringify(s1);

s1.name="Rishi";

console.log(s2);

//Task 4

function merge(a,b){

  var c={

    ...a,

    ...b

  }

  return c;

}

var a={

  name:"Rishi",

  id:123

}

var b={

  manager:"danush",

  mId:124

}

//console.log(Object.assign(a,b));

//console.log(merge(a,b));

// Task 5

var obj={

  n:"Rishi",

  id:33

}

var str=JSON.stringify(obj);

console.log(str);

var str1=JSON.parse(str);

console.log(str1);

// Task 1 Closure:

function funcOne(){

  let a=10;

  function funcTwo(){

    return a\*a\*4;

  }

  return funcTwo();

}

console.log(funcOne());

// Task 2

let c=10;

function counter(){

  function inc(){

    c=c+1;

    return c;

  }

  return inc();

}

console.log(counter());

console.log(counter());

console.log(counter());

// Task 3

let c1=1;

let g=1;

function counter(){

  function inc(){

    c1=c1+1;

    function inc1(){

      g=g+2;

      return g;

    }

    return inc1()+" "+c1;

  }

  return inc();

}

console.log(counter());

console.log(counter());

console.log(counter());

// Task 4

function p\_variables(){

    let a=0;

    function prv(){

        for(var i=0;i<=3;i++){

            a++;

            console.log(a);

        }

    }

    return prv();

}

p\_variables();

p\_variables();

// Task 5:

function factory(no) {

    for(var i=0;i<10;i++){

        no++;

    }

    return { no };

}

const obj1 = factory(4);

console.log(obj1.no);

//Task 1 Promise, Promises chaining:

function mynum(num){

  return new Promise((resolve) => {

    resolve(num\*60);

  });

}

console.log(mynum(4));

mynum(2).then(result=>console.log(result));

//task2

 new Promise(function(resolve,reject){

  setTimeout(()=>resolve(1),1000);

 }).then(function(result){

  return result\*2;

 }).then(function(result)

 {

  console.log(result);

  return result\*2;

 }).then(function(result)

 {

  console.log(result);

  return result\*2;

 })

// task4

let promise1 = new Promise(function (resolve, reject) {

  setTimeout(function () {

      resolve(1);

  }, 1000);

});

let promise2 = new Promise(function (resolve, reject) {

  setTimeout(function () {

      resolve(2);

  }, 2000);

});

Promise.all([promise1, promise2])

  .then(function (values) {

      console.log(values);

  })

// Task 3

var pro=new Promise((reject,resolve)=>{

  let a=0;

  if(a===0){

reject("no values");

  }else{

resolve("success");

  }

});

pro.then((message)=>{

    console.log(message);

}).catch((message)=>{

    console.log(message);

})