**S4\_\_Spread Operator And Functions**

* **Concat**:- used to concate two or more array
* **Spread Operator: (…)** : - it allows an iterable such as an array expression or string to be expanded in places where zero or more arguments (for function calls) or elements (for array literals) are expected, or an object expression to be expanded in places where zero or more key-value pairs (for object literals) are expected.
* **Destructuring:** The **destructuring assignment** syntax is a JavaScript expression that makes it possible to unpack values from arrays, or properties from objects, into distinct variables.
* **Functions:-** function keyword is used to declare function.its block of code. Only run when it calls. It contain return type.parameter non parameter. Used to perform certain action. Reusing code. ( ) used to define method. Method can be static non static, parameterise non parameterise.
* **Types: Parameterised Method :** the method contain some parameter in it.at the time of calling we need to pass **argument** for that method. Variables passed to method called **parameter**.
  + JavaScript function definitions do not specify data types for parameters.
  + JavaScript functions do not perform type checking on the passed arguments.
  + JavaScript functions do not check the number of arguments received.
* **Non Parameterised Method:** not contain any type of method.
* **Function Expression:-** A JavaScript function can also be defined using an **expression**. A function expression can be stored in a variable. Once the function stored in variable that variable is used as function. Its an **anonymous function** (a function without a name).Functions stored in variables do not need function names. They are always invoked (called) using the variable name.
* **Arrow Function in Javascript:-**  arrow function is easy to create. It’s a shorthand of function. ( )=> used to define.

**Scope of Function : Lexical Scope :-** a variable defined outside a function can be accessible inside another function defined after the variable declaration.

**Function Scope:** When a variable is declared inside a function, it is only accessible within that function and cannot be used outside that function.

Var is function scope. Can be used outside the block.

**Block Scope:** A variable when declared inside the if or switch conditions or inside for or while loops, are accessible within that particular condition or loop. Let and const is block scope. Cannot used outside the block.

**Global Block:-** used from anywhere in the file.

Program

let names = ["Keshav", "Kanchana", "Sahiba", "Imran", "Aniket", "Pritam", "Soham", "Ritu Raj"];

let cloneNames = [].concat(names);

console.log("Copied Names  " + cloneNames);

let cities = ["Pune", "Mumbai", "Panji"]

//Spread Operator

let spreadData = [...names, ...cities]; /// . accesing the value(Method,Object,Arrray) ... copying data from one to another

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

    console.log(spreadData[i]);

}

//Spread Operator in Object

const objectData = { "Name": "Sahiba", "Address": "Mumbai", "Email": "shb@123gmail.com", 100: 112, anotherObject: { "Name": "Afifa", "Address": "Pune", "Email": "afifa121@gmail.com", 100: 114 } };

const cloneObject = { ...objectData };

console.log(cloneObject)

console.log(cloneObject.anotherObject.Email) //Accessing VAlue

//Object Destructuring

let { Name, Address, ...restProps } = objectData;

console.log("Destruct Data:" + Name, Address, restProps);

// // Function

function welcome() {

    console.log("Welcome To Function World");

    console.log();

}

function arithmaticOperation() {

    welcome();

    var a = 40, b = 20, c;

    console.log(c = a + b);

    console.log(c = a \* b);

    console.log(c = a - b);

    console.log(c = a / b);

}

arithmaticOperation();

function displayName(names = []  /\*Parameter\*/) {

    console.log('Printing the Names');

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

        process.stdout.write(names[i] + "  ");

    }

}

// displayName(...names);

displayName(names /\*argument\*/); //Parameterised Function

//Function Expression

const palindromes = function(string) {

    const len = string.length;

    for (let i = 0; i < len / 2; i++) {

        if (string[i] !== string[len - 1 - i]) {

            return `${string} : It is not a palindrome`;

        }

    }

    return `${string} : It is a palindrome`;

}

const isEven = number => number % 2 == 0;

console.log("One Line Arrow Funcion: " + isEven(10));

const palindromes = (string) => {

    const len = string.length;

    for (let i = 0; i < len / 2; i++) {

        if (string[i] !== string[len - 1 - i]) {

            return `${string} : It is not a palindrome`;

        }

    }

    return `${string} : It is a palindrome`;

}

var prompt = require('prompt-sync')();

const palindrome = prompt('Enter a string or number: ');

console.log(palindromes(palindrome));

//Function Inside Function

function nestedFunction() {

    let sampleValue ="This is Main Variable";

    const addition = () => {

        // let sampleValue ="This is Nested Variable";

        var a = 100, b = 20, c = 40;

        var d=a+b+c;

        console.log("Inside the Function "+ sampleValue);

        console.log("Addition of 3 Number " +d);

    }

    const multiplication = function () {

        var a = 4, b = 8, c = 12;

        var d=a\*b\*c;

        console.log("Multiplication of 3 Number " + d);

    }

    const printingArray = function printing\_Array() {

        let names = ["Keshav", "Kanchana", "Sahiba", "Imran", "Aniket", "Pritam", "Soham", "Ritu Raj"];

        return names;

    }

    console.log("Main Function "+ sampleValue);

    addition();

    multiplication();

    console.log("Names Are : " +printingArray());

}

console.log("Calling Main Function");

nestedFunction();

// Q]] Write a Program to check Armstrong Palindrome Even Odd Number Using All

        //All : Switch If For Function

//Q]] Write Program to Copy object data into Array or Vise versa?