**1.DATATYPES IN JAVASCRIPT?**

**THERE ARE TWO DATATYPES IN JAVACRIPT**

**1.PRIMITIVE DATATYPES**

**2.NON-PRIMITIVE DATATYPES**

**PRIMITIVE DATATYPES**

**NUMBER**

**STRING**

**BOOLEAN**

**NULL**

**UNDEFINED**

**SYMBOL**

**BIG INT**

**NON-PRIMITIVE DATATYPES**

**OBJECT**

**2.NUMBER**

**JAVASCRIPT DOES NOT DEFINE DIFFERENT TYPES OF NUMBERS,LIKE INTEGERS,SHORT,LONG,FLOATING-POINT ETC.**

**EX:**

**25**

**5.3**

**123e5(12300000)**

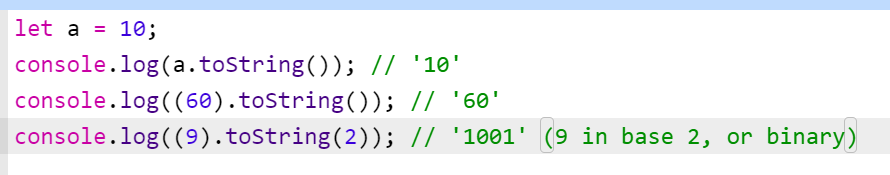
**123e-5(0.00123)**

**Infinity**

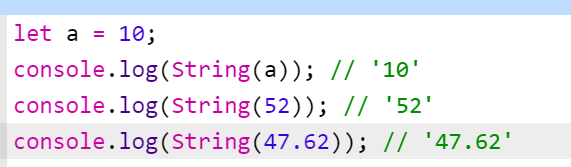
**-Infinity**

**3.HOW TO CONVERT NUMBER TO STRING USING JAVASCRIPT**

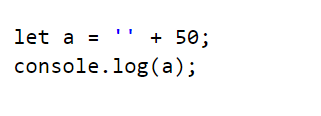
**WE CAN USE toString () METHOD.**

****

**WE CAN USE String () METHOD ALSO.**

****

**CONCATANATE EMPTY STRING**

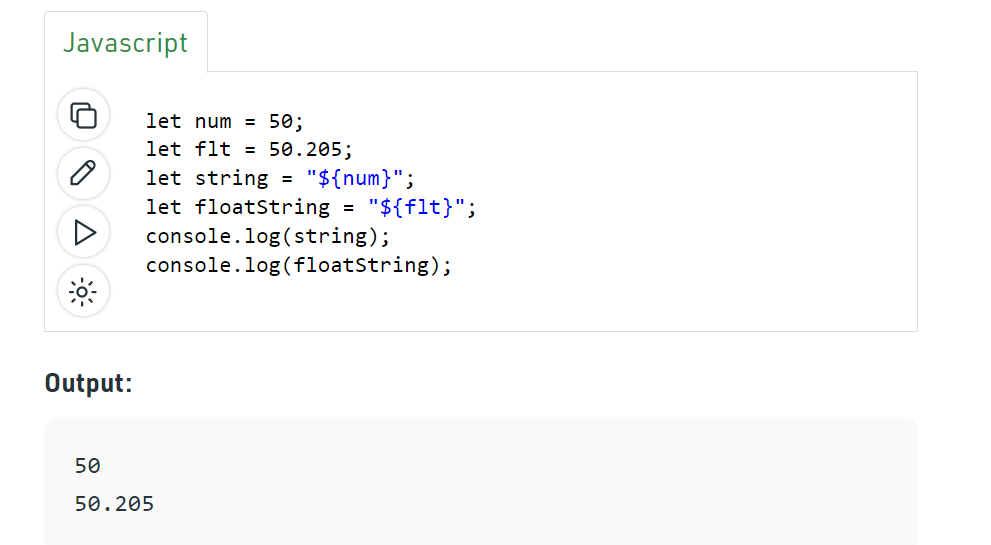
****

**OUTPUT:**

****

**USING TEMPLATE STRINGS**

**INJECTING NUMBER INSIDE STRING**

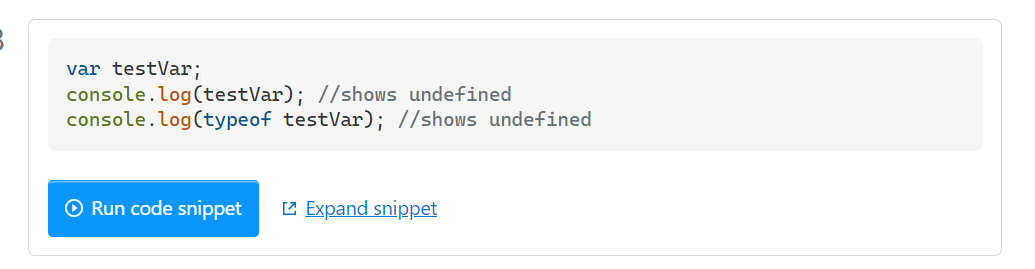
****

**4.UNDEFINED VS NULL**

**UNDEFINED**

**“UNDEFINED” MEANS VARIABLE DECLARED BUT NOT ASSIGNED VALUE.**

**“Typeof UNDEFINED” IS UNDEFINED.**

****

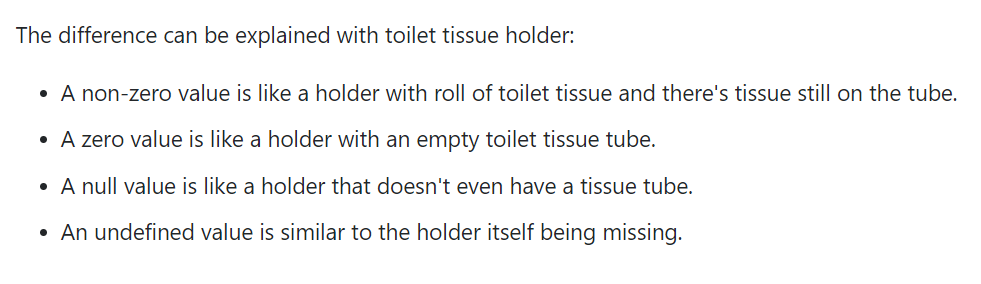
**NULL**

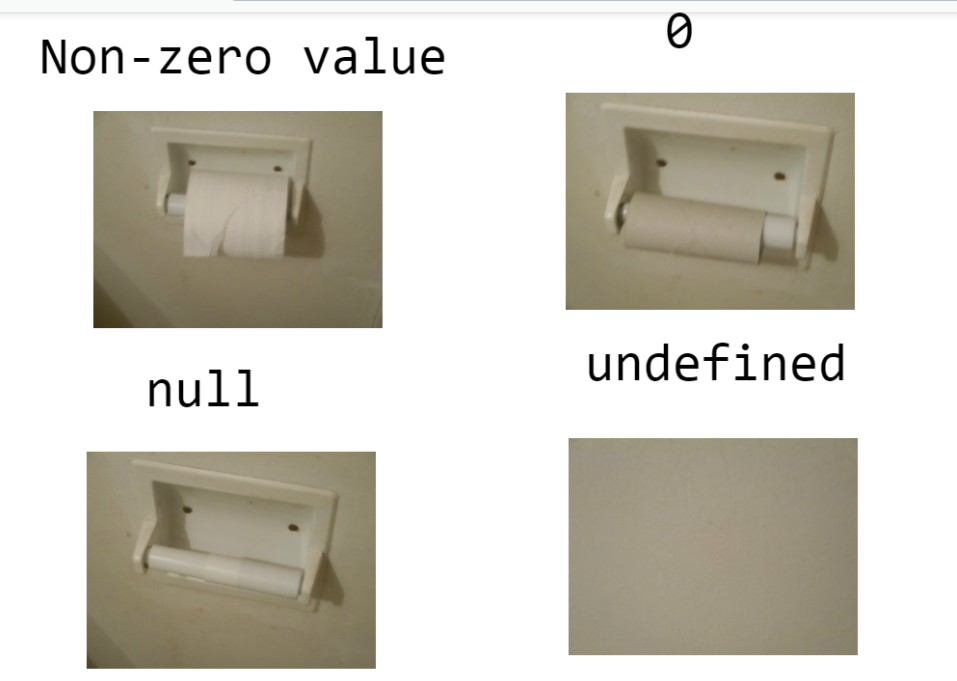
**NULL IS ASSIGNMENT VALUE.IT CAN BE ASSIGNED TO VARIABLE AS REPRESENTATION OF NO VALUE.**

**“typeof NULL” IS OBJECT.**

**Graphical user interface, text, application

Description automatically generated**

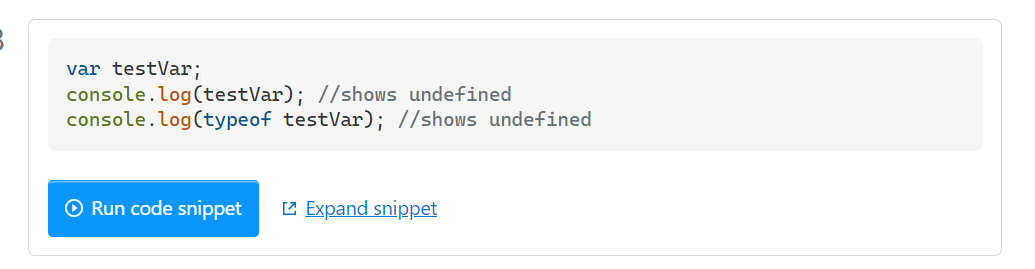
****

****

**5.UNDEFINED VS NOT DEFINED**

**UNDEFINED**

**“UNDEFINED” MEANS VARIABLE DECLARED BUT NOT ASSIGNED VALUE.**

****

**NOTDEFINED**

**NEITHER DECLARED AND NOR DEFINED THAT IS CALLED NOT DEFINED.**

**EX:**

**console.log(x)**

**6.STRING EXAMPLES**

**“23”**

**“null”**

**“undefined”**

**“5.3”**

**“true”**

**“false”**

**7.BOOLEAN EXAMPLES**

**TRUE OR FALSE**

**8.PRIMITIVE VS NON-PRIMITIVE DATA TYPE**

**GO THROUGH BELOW URL**

**https://dmitripavlutin.com/value-vs-reference-javascript/#:~:text=In%20JavaScript%2C%20you%20can%20pass,by%20reference%20when%20assigning%20objects**

**9.== VS ===**

**“==” COMPARES VALUE ONLY**

**“===” COMPARES VALUE AND DATATYPE.**

EXAMPLES

=========

**FALSE==’0’ //TRUE**

**FALSE===’0’//FALSE**

**UNDEFINED==NULL//TRUE**

**UNDEFINED===NULL//FALSE**

**[]==[]//FALSE**

**[]===[]//FALSE**

**{}=={}//FALSE**

**{}==={}//FALSE**

**10.’typeof’ OPERATOR**

**’typeof’ OPERATOR IS USED TO RETURN STRING DESCRIPTION OF THE TYPE OF VARIABLE.**

**typeof 42 //number**

**typeof 2.3 //number**

**typeof infinity // number**

**typeof ‘html’ //string**

**typeof true //boolean**

**typeof false //boolean**

**typeof undefined //undefined**

**typeof function //function**

**typeof null //object**

**typeof bigint //bigint**

**typeof array //object**

**typeof object //object**

**typeof (65) //number**

**typeof Math.LN10 //number**

**typeof NaN //number**

**typeof ‘1’ //string**

**typeof “ ” //string**

**typeof (typeof 1) //string**

**typeof (String(1)) //string**

**typeof `hello` //string**

**typeof !!(1) //boolean**

**typeof Symbol() //symbol**

**typeof Symbol(‘party’) //symbol**

**typeof Symbol.iterator //symbol**

**typeof {b:1} //object**

**typeof [] //object**

**typeof {} //object**

**typeof [1,2,9] //object**

**typeof new Date() //object**

**typeof function (){} //function**

**typeof class C{} //function**

**typeof Math. sin //function**

**11.WHAT IS “isNaN”**

**2.ES6 NEW FEATURES**

ARROW FUNCTION

SPREAD AND REST OPERATOR

TEMPLATE LITERALS

ARRAY AND OBJECT DESTRUCTURING

DEFAULT PARAMETERS

CLASSES

LET AND CONST

PROMISES

**3.SCOPE**

|  |  |  |  |
| --- | --- | --- | --- |
| GLOBAL | LOCAL | BLOCK | LEXICAL |
| VARIABLES DECALRED OUTSIDE ANY FUNCTION HAVE GLOBAL SCOPE.  GLOBAL VARIABLES CAN BE ACCESSED FROM ANYWHERE IN JAVASCRIPT PROGRAM. | LOCAL SCOPE OR FUNCTION SCOPE MEANS VARIABLE DEFINED INSIDE FUNCTION. | BLOCK SCOPE MEANS VARIABLE DEFINED INSIDE BLOCK.{} | LEXICAL SCOPE MEANS WHENEVER CODE TRIES TO ACCESS VARIABLE DURING FUNCTION CALL IT STARTS SEARCHING FROM LOCAL VARIABLES.IF THE VARIABLE IS NOT FOUND IT WILL CONTINUE SEARCHING OUTER SCOPE OR PARENT FUNCTION SCOPE UNTIL REACHES GLOBAL SCOPE. |

**4.VAR, LET, CONST**

|  |  |  |
| --- | --- | --- |
| **VAR** | **LET** | **CONST** |
| ‘VAR’ IS THERE IN JS FROM THE BEGINNING. | ‘LET AND CONST’ WAS INTRODUCED IN ES6 | |
| USING ‘VAR’ WE CAN RE-DECLARE AND UPDATE | USING ‘LET’ WE CANNOT RE-DECLARE BUT WE CAN UPDATE | USING ‘CONST’ WE CANNOT RE-DECLARE AND WE CANNOT UPDATE |
| ‘VAR’ HAS TWO SCOPES  LOCAL AND GLOBAL | ‘LET’ HAS THREE SCOPES. GLOBAL, LOCAL,BLOCK | ‘CONST’ HAS THREE SCOPES. GLOBAL,LOCAL,BLOCK |
| ‘VAR’ HOSTING WILL HAPPEN | ‘LET AND CONST’ ALSO HOSTING WILL HAPPEN BUT NOT INITIALIZED WITH UNDEFINED. | |

**5.TEMPLATE LITERALS**

**TEMPLATE LITERALS DINOTED BY BACKTICKS (`).**

**IT ALLOWS MULTILINE STRINGS,VARIABLE SUBSTITUTION AND EXPRESSION SUBSTITUTION**

**MULTILINE STRING EXAMPLE**

let x=`hyper

text

markup`

console.log(x);

**VARIABLE SUBSTITUTION EXAMPLE**

let x=’hello’;

let y=’good morning’;

let wish=`${x} ${y}`;

console.log(wish);

output: hello good morning

**EXPRESSION SUBSTITUTION EXAMPLE**

let x=2;

let y=3;

let result=`${x+y}``;

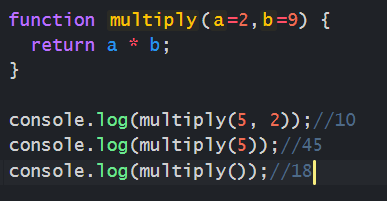
console.log(result);

output: 5

**6.DEFAULT PARAMETERS IN JAVASCRIPT**

DEFAULT PARAMETERS ALLOW PARAMETERS TO BE INITILIZED WITH DEFAULT VALUES,IF “NO VALUE” OR “UNDEFINED” PASSED.

**EXAMPLE:**

****

**51.DIFFRENT WAYS TO DECLARE FUNCTIONS?**

* **FUNCTION DECLARATION**
* **FUNCTION EXPRESSION**
* **ANONOYMUS FUNCTION**
* **ARROW FUNCTION**
* **IIFE (IMMEDIATE INVOKE FUNCTION EXPRESSION)**
* **GENERATORS**

3.STRING METHODS

SLICE

SUBSTR

SUBSTRING

REPLACE

TOUPPERCASE

TOLOWERCASE

CONCAT

CHARAT

CHARCODEAT

SPLIT

PADSTART

PADEND

4. SPLICE

* USING “SPLICE” METHOD WE CAN REMOVE ELEMENTS AND ADD ELEMENTS FROM ARRAY.
* IT WILL TAKE 3 PARAMETERS
* FIRST PARAMETER DEFINES WHERE NEW ELEMENTS SHOULD ADDED OR REMOVED
* SECOND PARAMETER DEFINES HOW MANY ELEMENTS SHOULD REMOVED
* THIRD PARAMETER DEFINED HOW MANY ELEMENTS SHOULD BE ADDED.
* IT WILL CHANGE ORIGINAL ARRAY

5.SLICE

* SLICE METHOD EXTRACT ELEMENTS OF THE ARRAY
* IT DOES NOT CHANGE ORIGINAL ARRAY.IT WILL CREATE NEW ARRAY
* IT WILL TAKE TWO PARAMETERS. BOTH ARE OPTIONAL

7.DISPLAY POSSIBILITIES?

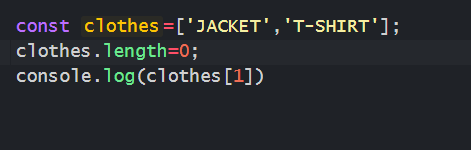
document.write

window.alert

console.log

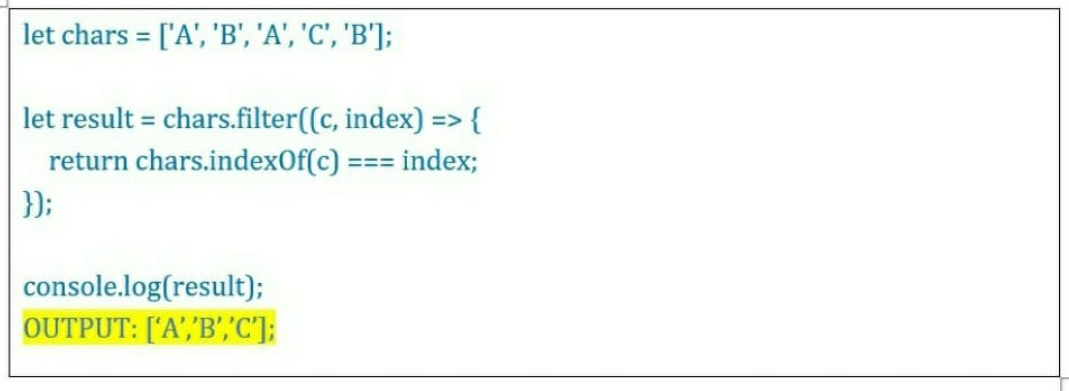
innerHTML

**8.SNIPPET**

****

**OUTPUT: UNDEFINED**

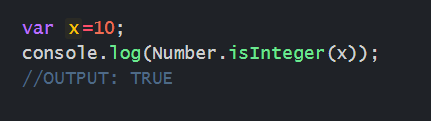
**9.REMOVE DUPLICATES FROM ARRAY USING JAVASCRIPT?**

****

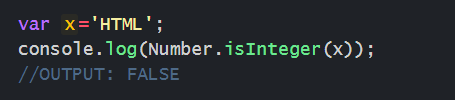
**10.HOW TO CHECK GIVEN NUMBER INTEGER OR NOT USING JAVASCRIPT?**

**WE CAN USE “Number.isInteger” method.**

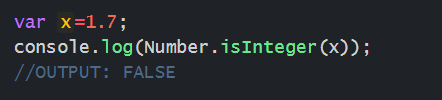
**EX 1:**

****

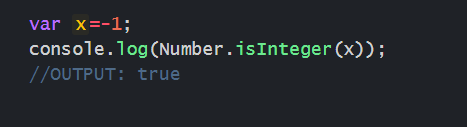
**EX 2:**

****

**EX 3:**

****

**EX 4:**

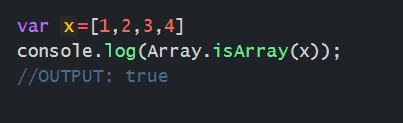
****

**NOTE:** INTEGERS ARE NUMBERS THAT CAN BE POSITIVE, NEGATIVE OR ZERO. BUT CANNOT BE FRACTION.

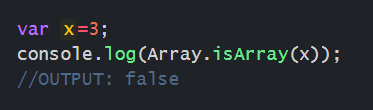
**11.HOW TO CHECK VARIABLE IS ARRAY OR NOT USING JAVASCRIPT?**

**WE CAN USE “Array.isArray()” METHOD.**

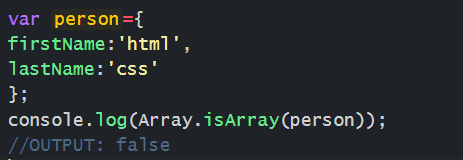
**EX 1:**

****

**EX 2:**

****

**EX 3:**

****

**12.HOW MANY WAYS WE CAN CREATE OBJECTS IN JAVASCRIPT?**

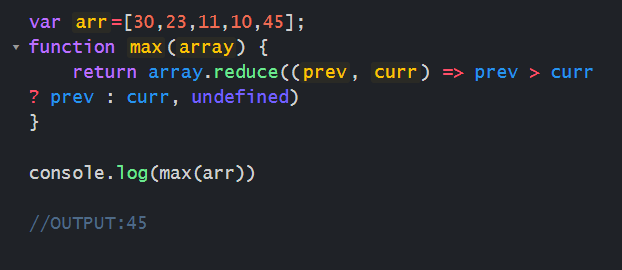
**1.USING OBJECT LITERALS**

**2.USING OBJECT.CREATE**

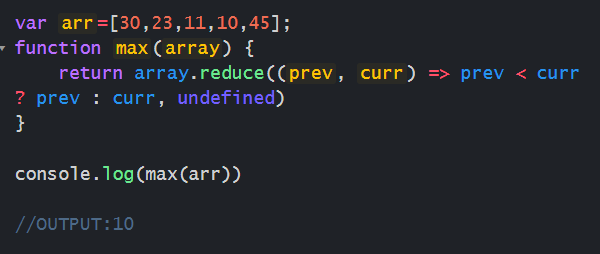
**3.USING ES6 CLASSES**

**4.USING CONSTRUCTOR**

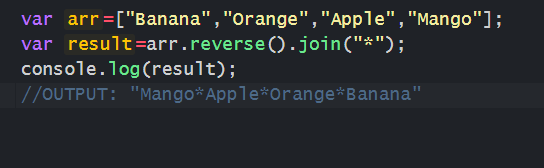
**13.HOW TO FIND MAXIMUM NUMBER WITHOUT BUILT-IN METHOD.**

****

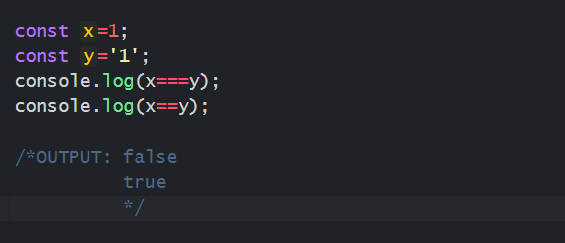
**14.HOW TO FIND MINIMUM NUMBER WITHOUT BUILT-IN METHOD**

****

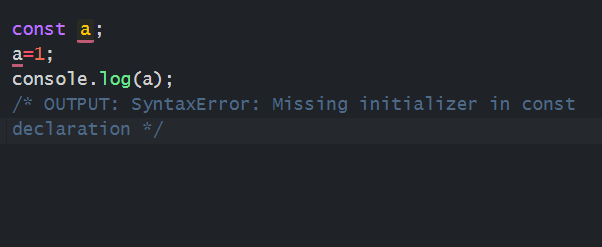
**15.EXPECTED OUTPUT “Mango\*Apple\*Orange\*Banana”**

****

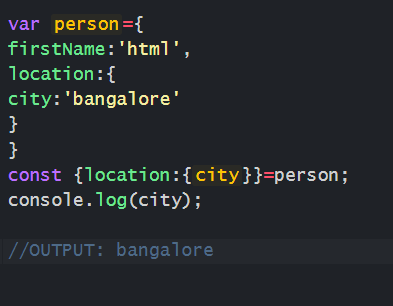
**16.SNIPPENT**

****

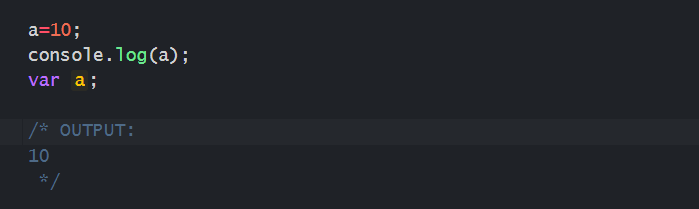
17.SNIPPET



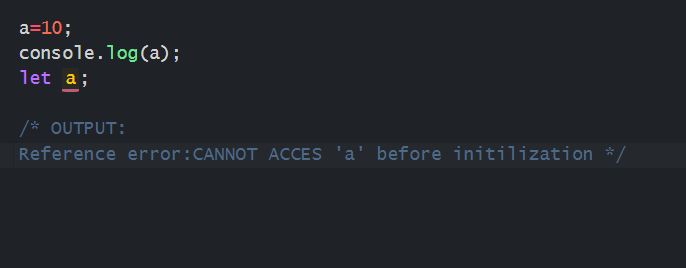
18.NESTED OBJECT DESTRUCTURING EXAMPLE?



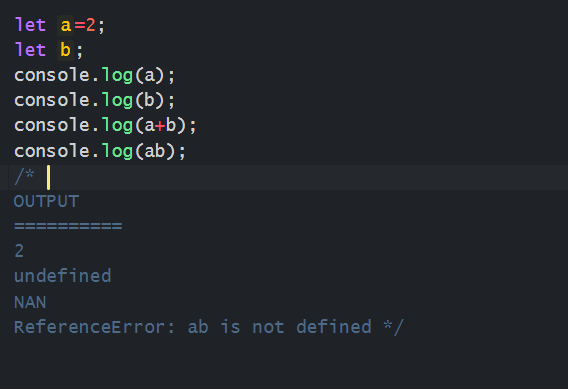
19.SNIPPET



20. SNIPPET



21.SNIPPET



22.DIFFERENCE BETWEEN NULL AND UNDEFINED?

**UNDEFINED:** DECLARED A VARIBLE WITHOUT ASSIGNING VALUE THAT IS DEFINED.

Ex: var x;

**NULL:** NULL IS ASSIGNMENT VALUE AS A REPRESANTATION OF NO VALUE.

EX: var x=null;

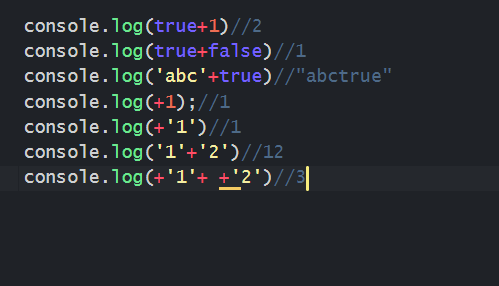
23.HOSTING

HOSTING IS JAVASCRIPT DEFAULT BEHAVIOUR, MOVING ALL DECLARATION TO THE TOP OF THE CURRENT SCOPE THAT IS CALLED HOSTING.

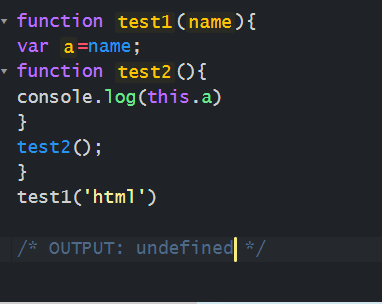
24.JAVASCRIPT IS SINGLE THREAD OR MULTI THREAD?

SINGLETHREAD

25.SNIPPET

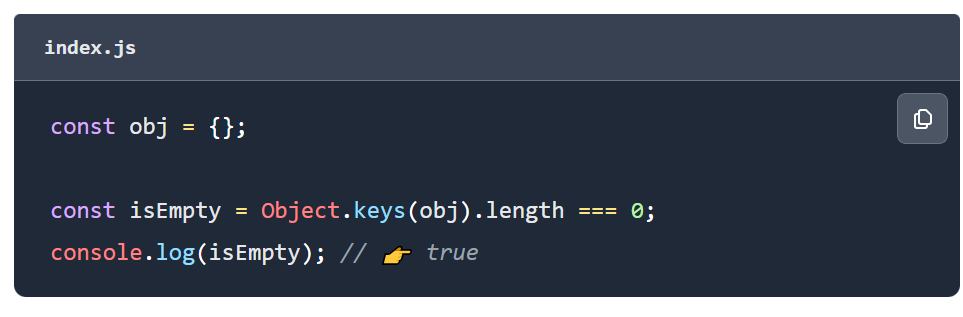


26.SNIPPET



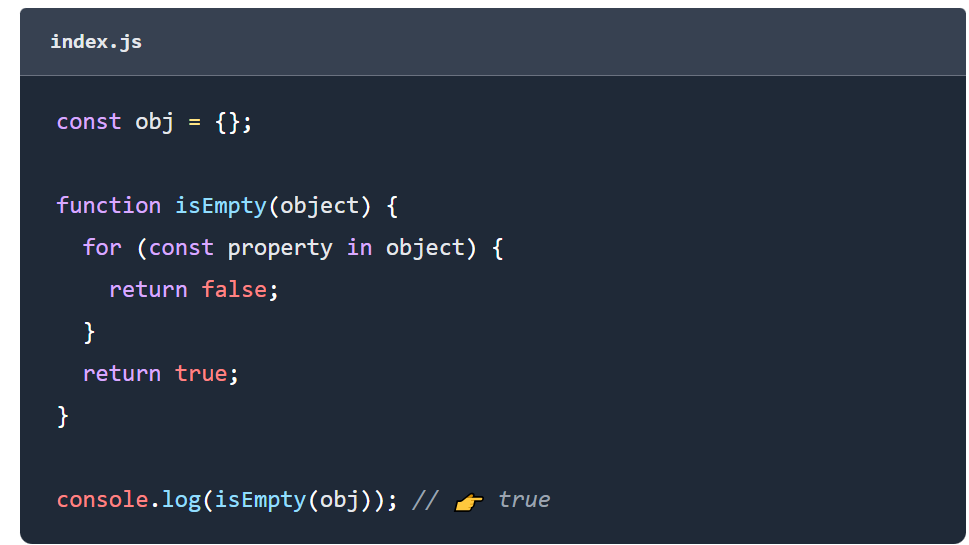
27.HOW TO CHECK OBJECT EMPTY OR NOT USING JAVASCRIPT?

**USING OBJECT KEYS**





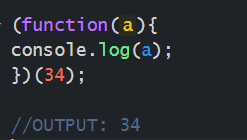
**USING FOR IN LOOP**

****

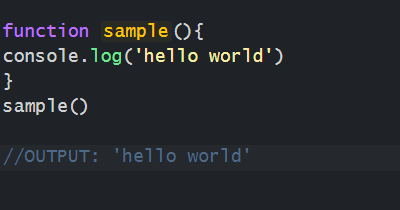
**28.IIFE**

* **IMMEDIATE INVOKE FUNCTION EXECUTION**
* **IIFE WON’T STORE GLOBALLALY**

****

****

**29.NAMED FUNCTION OR FUNCTION DECLARATION**

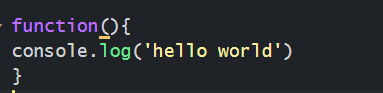
****

**NOTE:**

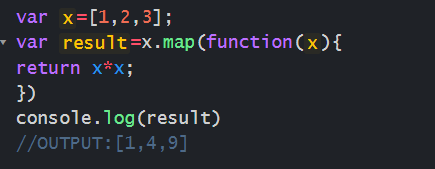
* **NAMED FUNCITONS AND ANONYMOUS FUNCTIONS WON’T GET CALLED UNTIL WE MAKE CALL.**
* **THIS FUNCTIONS WILL BE STORED GLOBALLY, WE CAN RE-USE LATER.**

**30.ANONYMOUS FUNCTION**

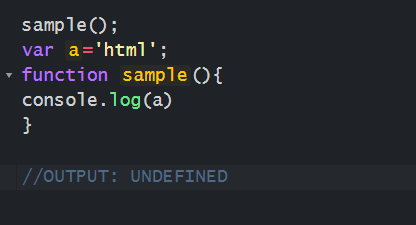
**ANONYMOUS FUNCTION MEANS WE USE ONLY FUNCTION KEYWORD WITHOUT FUNCTION NAME.**

****

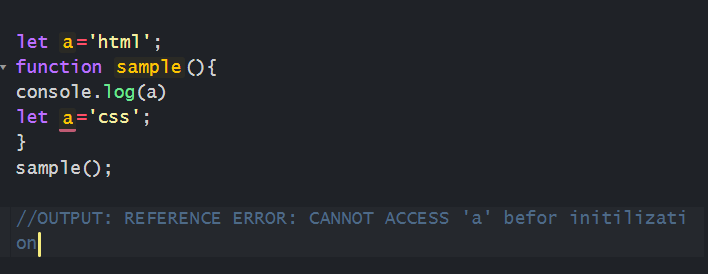
**WE CAN ALSO PASS ANONYMOUS FUNCTION AS A PARAMETER TO ANOTHER FUNCTION.**

****

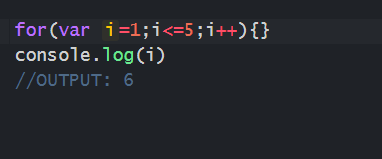
**31.SNIPPET**

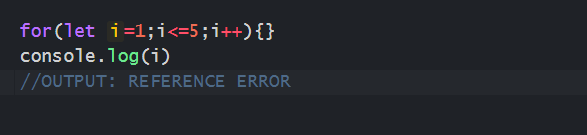
****

**32.SNIPPET**

****

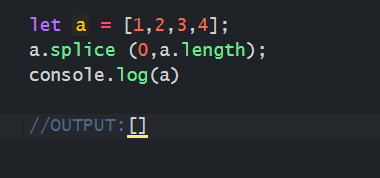
**33.SNIPPET**

****

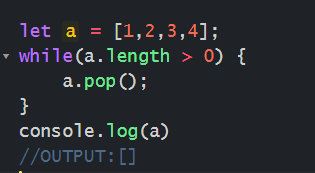
**34. SNIPPET**

**35.HOW TO EMPTY ARRAY USING JAVACRIPT**

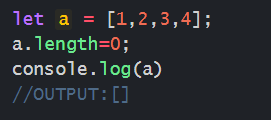
**USING SPLICE**



USING POP METHOD

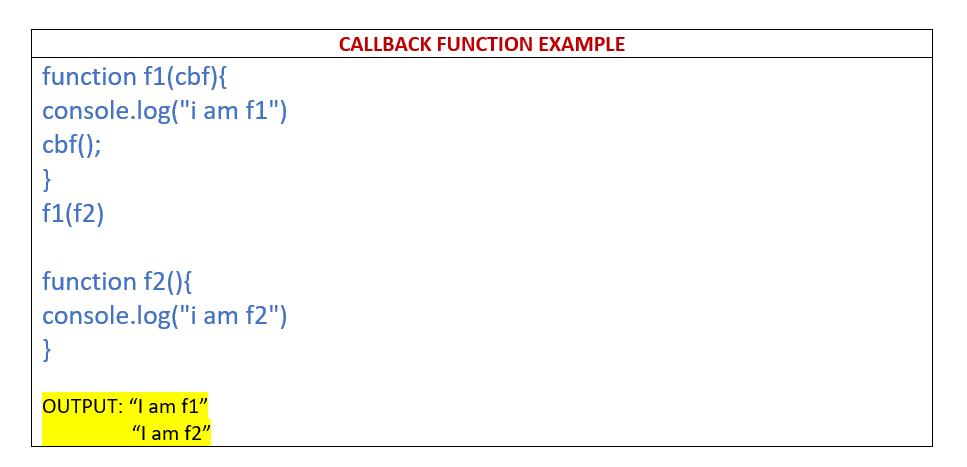
****

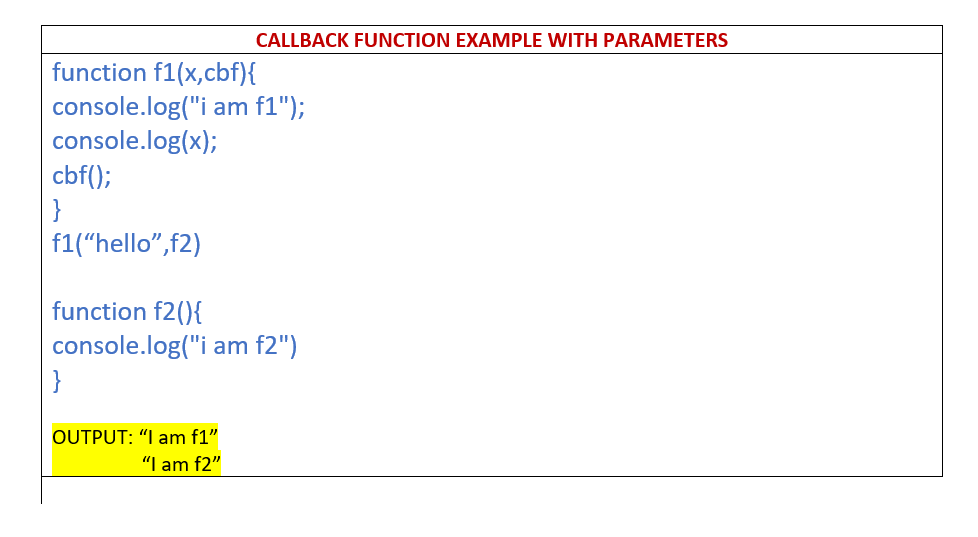
SETTING ITS LENGTH TO ZERO



**36.WHAT IS CALLBACK FUNCTION**

PASSING A FUNCTION AS AN ARGUMENT TO ANOTHER FUNCTION THAT IS CALLBACK FUNCTION.

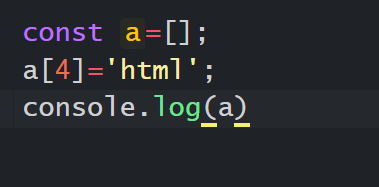




37.HOW MANY DATATYPES ARE THERE?

8

38.SNIPPET

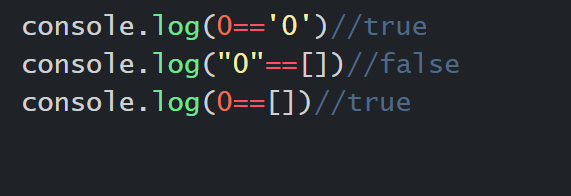


OUTPUT:

****

**LENGTH OF ‘a’ IS 5.**

**39.SNIPPET**

****

**40.EVENTS IN JAVASCRIPT**

**onclick**

**ondblclick**

**onchange**

**onfocus**

**onblur**

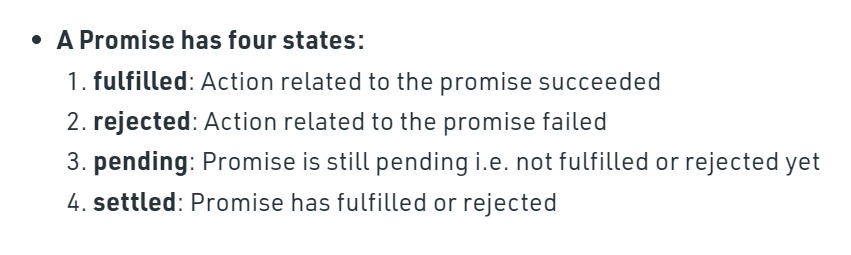
**onload**

**onsubmit**

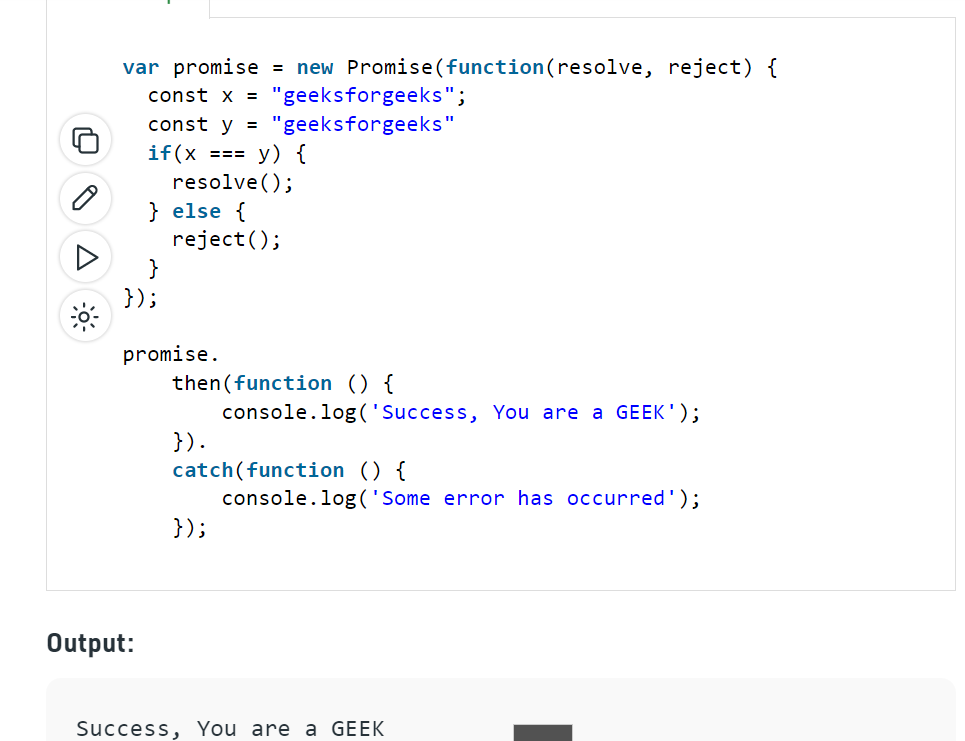
**onreset**

**41.PROMISES**

* **PROMISES ARE USED TO HANDLE MULTIPLE ASYNCHRONOUS OPERATIONS IN JAVASCRIPT**
* **MULTIPLE CALLBACK FUNCTIONS WOULD CREATE CALLBACK HELL. TO AVOID WE CAN USE PROMISES**
* **PROMIS BENFITS ARE, IMPROVES CODE READLITY AND BETTER ERROR HANDLING**
* **PROMISE HAS FOUR STATES,**

****

* **PROMISE TAKES ONLY ONE ARGUMENT WHICH IS CALLBACK FUNCTION.**
* **THAT CALLBACK FUNCTION TAKES TWO ARGUMENTS 1.RESOLVE 2.REJECT**
* **IF PROMISE RESOLVED IT WILL GO TO THE “THEN” BLOCK.**
* **IF PROMISE REJECTED IT WILL GO TO THE “CATCH” BLOCK.**

****

**42.ARGUMENT VS PARAMETERS**

**FUNCTION PARAMETERS ARE LISTED WHEN FUNCTION IS DECLARED.**

**FUNCTION ARGUMENTS ARE PASSED WHEN THE FUNCTION IS CALLED**

**43.HIGH ORDER FUNCTIONS**

**HIGH ORDER FUNCTION IS A FUNCTION THAT RECIEVES FUNCTION AS AN ARGUMENT.**

**MAP**

**REDUCE**

**FILTER**

**FOREACH**

**44.SCOPE CHAIN IN JAVASCRIPT**

**WHEN EVER CODE TRIES TO ACCESS VARIABLE DURING FUNCTION CALL IT STARTS SEARCHING FROM LOCAL VARIABLES.IF THE VARIABLE IS NOT FOUND IT WILL CONTINUE SEARCHING OUTER SCOPE OR PARENT FUNCTION SCOPE UNTIL REACHES GLOBAL SCOPE.**

**45.ARRAY METHODS**

**toString()**

**join()**

**push()**

**pop()**

**shift()**

**unshift()**

**splice()**

**slice()**

**46.POP UP BOXES AVAILABLE IN JAVASCRIPT**

**ALERT BOX**

**CONFIRM BOX**

**PROMPT BOX**

**47.TYPE OF OPERATOR**

**’TYPE OF’ OPERATOR IS USED TO RETURN STRING DESCRIPTION OF THE TYPE OF VARIABLE.**

**typeof 42 //number**

**typeof 2.3 //number**

**typeof infinity // number**

**typeof ‘html’ //string**

**typeof true //boolean**

**typeof false //boolean**

**typeof undefined //undefined**

**typeof function //function**

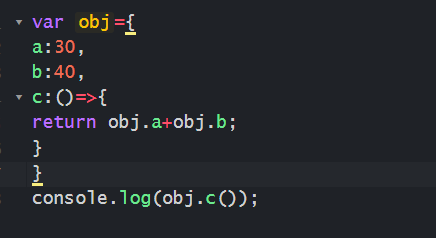
**typeof null //object**

**typeof bigint //bigint**

**typeof array //object**

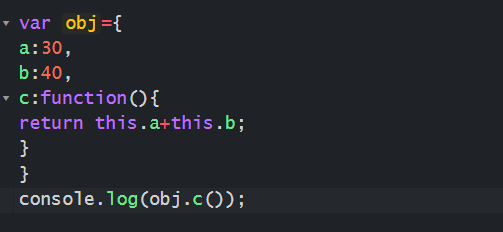
**typeof object //object**

**48.ARROW FUNCTION AS OBJECT METHOD**

****

**OUTPUT 70**

**49.NORMAL FUNCTION AS OBJECT METHOD**

****

**OUTPUT 70**

**50.LEXICAL SCOPE**

**When ever code tries to access variable during function call it starts searching from local variables.if the variable is not found it will continue searching outer scope or parent function scope until reaches global scope.**

**But the opposite is not true, the variables defined inside function will not be accessible outside function**

|  |  |  |  |
| --- | --- | --- | --- |
| Example: | Example: | Example | Example |
| function parent () {  var course='html'  function child(){  var course='js'  function grandchild(){  var course='css'  console.log(course)  }  grandchild();  }  child();  }  parent(); | function parent () {  var course='html'  function child(){  var course='js'  function grandchild (){  console.log(course)  }  grandchild();  }  child();  }  parent(); | function parent () {  var course='html'  function child(){  function grandchild (){  console.log(course)  }  grandchild();  }  child();  }  parent(); | function parent () {  console.log(course)  function child(){  var course='js'  }  child();  }  parent(); |
| Output: ‘css’ | Output: ‘js’ | Output:’html’ | Output:Reference error:course is not defined |

**52.REST OPERATOR**

**The rest operator allows us to call a function with number of arguments. It is denoted by three dots(…).**

|  |  |
| --- | --- |
| **Without rest operator** | **With rest operator** |
| **function fun(a, b){**  **return a + b;**  **}**  **console.log(fun(1, 2));**  **console.log(fun(1, 2, 3, 4, 5));** | **function fun(a,b,...c){**  **console.log(`${a} ${b}`);**  **console.log(c);**  **console.log(c[0]);**  **console.log(c.length);**  **console.log(c.indexOf('Lionel'));**  **}**  **fun('Mukul','Latiyan','Lionel','Messi','Barcelona');** |
| **Output: 3**  **3** | **Output:**  **Mukul Latiyan**  **[“Lionel”, “Messi”,” Barcelona”] Lionel 3 0** |

|  |
| --- |
| **Sum of numbers using rest operator** |
| **function** sum(...args){     let total = 0;  **for**(let number of args){        total += number;     }  **return** total;  }  console.log(sum(1, 2));  console.log(sum(1, 2, 3)); |
| **Output: 3**  **6** |

**53.WHAT IS DOM?**

**dom stands for document object model. when webpage is loaded , the browser creates a dom. dom is constructed as tree of objects.**

**54.SNIPPET**

**function x(){**

**setTimeout(function(){**

**console.log(i)**

**},1000)**

**var i=10;**

**}**

**x()**

**OUTPUT:10**

**55.DATATYPE OF “THIS”**

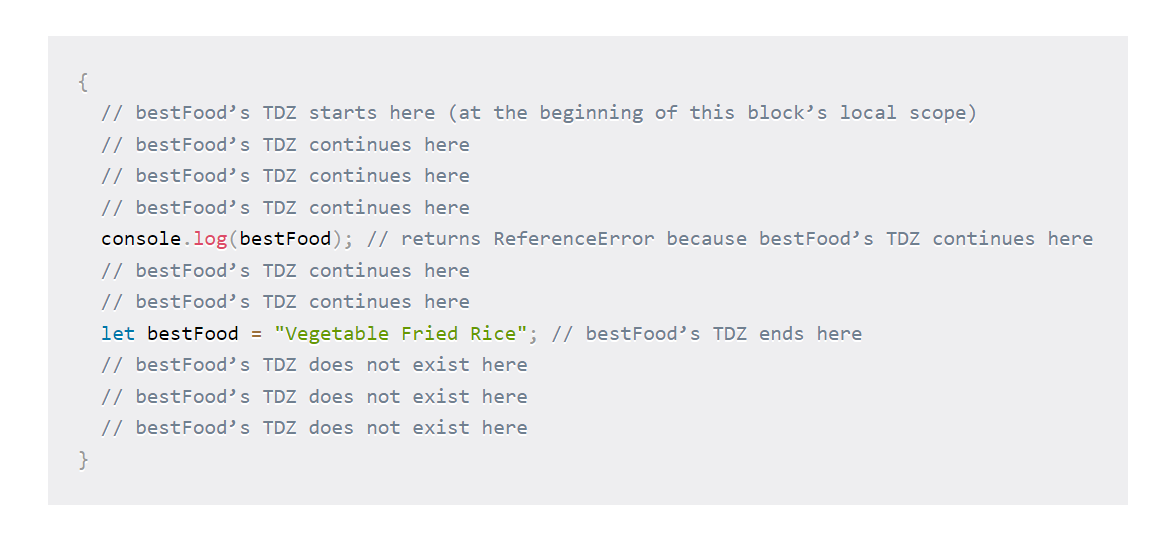
**Window**

**56.this===window**

**True**

**57.TEMPORAL DEAD ZONE?**

**Let and const variables hosted.but there is period between entering scope and declared variables they cannot be accessed. This is temporal dead zone.**

****

**58.JAVASCRIPT IS SYNCHRONOUS OR ASYNCHRONOUS?**

**SYNCHRONOUS**

**59.’THIS’ KEYWORD IN JAVACRIPT**

**‘THIS’ KEYWORD REFERS DIFFERENT OBJECTS**

* **IN OBJECT METHOD,”THIS” REFERS TO THE OBJECT**
* **ALONE “THIS” REFERS TO THE GLOBAL OBJECT**
* **IN FUNCTION THIS REFERS TO THE GLOBAL OBJECT**
* **IN FUNCTION,IN STRICTMODE “THIS” IS UNDEFINED.**
* **CALL,APPLY,BIND THIS REFERS TO THE ANY OBJECT**

**60.HOW TO HANDLE ASYNCRONOUS OPERATIONS IN JAVASCRIPT?**

**WE CAN HANDLE USING CALLBACK,PROMISES AND ASYNC/AWAIT**

**61.WHAT IS SPREAD OPERTAOR?**

**SPREAD OPERATOR DENOTED BY (…).**

**USING SPREAD OPERATOR WE CAN USE.**

**\*MERGE ARRAYS AND OBJECTS**

**\*EXPAND ARRAYS AND OBJECTS**

**\*CONVERT ARRAY TO ARGUMENTS**

**\*COPYING ARRAYS AND OBJECTS**

**62.CALL,APPLY,BIND**

* **CALL:USING CALL() METHOD, WE CAN WRITE A METHOD THAT CAN BE USED IN DIFFERENT OBJECTS**
* **APPLY: SAME AS CALL METHOD IT WILL TAKE ARGUMENTS AS AN ARRAY**
* **BIND: SAME AS CALL METHOD, BUT ONLY DIFFRENCE IS BIND KEEPS A COPY OF THAT METHOD. INVOKES LATER.**

**63.DESTRUCTURING**

**USING DESTRUCTURING WE CAN EASILY UNPACK THE VALUES FROM ARRAY AND OBJECTS.**

**64.WHAT IS ARROW FUNCTION**

* **ARROW FUNCTIONS INTRODUCED IN ES6**
* **IF FUNCTION HAS ONLY ONE ARGUMENT, YOU CAN OMIT THE PARANTHSIS.**
* **IF FUNCTION HAS ONLY ONE STATEMENT YOU CAN REMOVE BRACKETS AND RETURN KEYWORD.**
* **WITH ARROW FUNCTION “THIS” KEYWORD ALWAYS REFERS WINDOW OBJECT.**
* **ARROW FUNCTION DON’T HAVE ARGUMENTS.**
* **ARROW FUNCTION NOT SUITABLE FOR CALL ,APPLY,BIND**
* **ARROW FUNCTION NOT SUITABLE FOR CONSTRUCTORS**

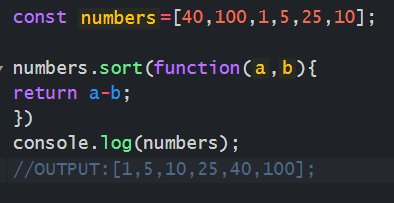
**65.WHAT IS WEBSOCKET**

**THE “WEBSOCKET” PROTOCOL, PROVIDES WAY EXCHANGE DATA BETWEEN BROWSERS AND SERVER VIA PERSISTENT CONNECTION. DATA CAN BE PASSED IN BOTH DIRECTIONS AS PACKETS WITHOUT BREAKING THE CONNECTION.**

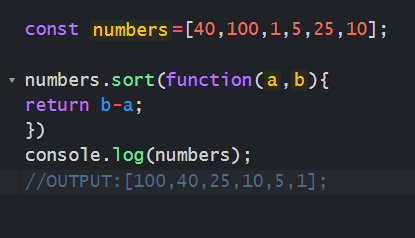
**66.WHAT IS PROTOTYPE?**

**PROTOTYPE IS AN OBJECT.WHERE WE CAN ATTACH METHODS AND PROPERTIES IN PROTOTYPE.**

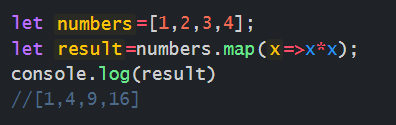
**67.SORT ARRAY(ASSENDING ORDER)**

****

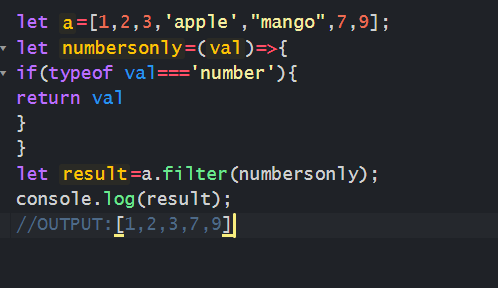
**67.SORT ARRAY (DECENDING ORDER)**

****

**68.SQARE EACH NUMBER IN ARRAY JAVASCRIPT**

****

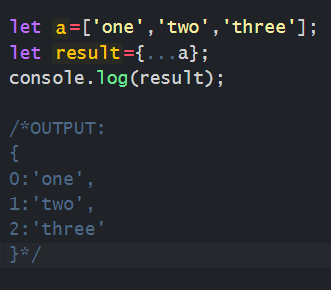
**69.FILTER NUMBERS ONLY IN ARRAY?**

****

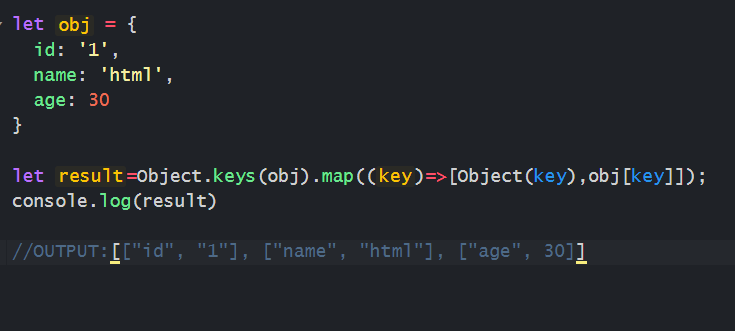
**70.DIFFERENCE BETWEEN MAP VS FOREACH?**

* **MAP RETURNS NEW ARRAY**
* **FOREACH RETURNS UNDEFINED**
* **MAP IT WILL SUPPORT METHOD CHAINING**
* **FOREACH IN WON’T SUPPORT METHOD CHAINING**

**71.CONVERT ARRAY INTO OBJECT**

****

**72.CONVERT OBJECT TO ARRAY?**

****

**73.OBJECT.KEYS**

**“Object.keys()” method returns**

**74.ATOMIC DESIGN PATTERN**

**ATOMIC DESIGN PATTERN HELPS BUILD CONSISTENT,SOLID RE-USABLE DESIGN SYSTEMS.**

**ATOMIC DESIGN COMES FROM THE IDEA OF SEPARATING THE COMPONENTS,TEMPLATES,PAGES.**

**75.”FIND” METHOD?**

* **“FIND” METHOD RETURNS THE VALUE OF THE FIRST ELEMENT THAT PASSES THE TEST.**
* **IT EXECUTES A FUNCTION FOR EACH AND ARRAY ELEMENT**
* **IT RETURNS UNDEFINED IF NO ELEMENTS ARE FOUND**
* **IT DOES NOT EXECUTE THE FUNCTION FOR EMPTY ELEMENTS**
* **IT DOES NOT CHANGE ORIGINAL ARRAY.**

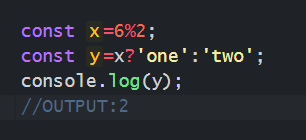
**76.”SOME” METHOD?**

**“SOME” METHOD RETURNS TRUE. IF THE FUNCTION RETURNS TRUE FOR ONE OF THE ARRAY ELEMENTS.**

**“SOME” METHOD RETURNS FALSE. IF THE FUNCTION RETURNS FALSE FOR ALL OF THE ARRAY ELEMENTS.**

**IT DOES NOT CHANGE ORIGINAL ARRAY.**

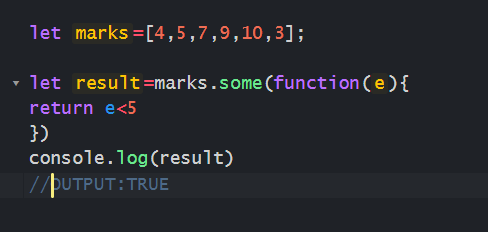
**77.SNIPPET**

****

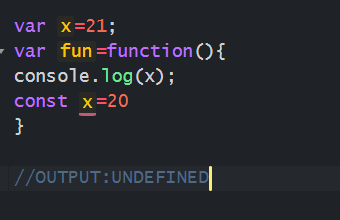
**78.SNIPPET**

****

**79. SNIPPET**

****

**80.SNIPPET**

****

**81.SORT ARRAY OF OBJECTS BASED ON AGE.**

|  |
| --- |
| **let emp = [**  **{**  **firstName: 'John',**  **lastName: 'Doe',**  **age: 27,**  **joinedDate: 'December 15, 2017'**  **},**  **{**  **firstName: 'Ana',**  **lastName: 'Rosy',**  **age: 25,**  **joinedDate: 'January 15, 2019'**  **},**  **{**  **firstName: 'Zion',**  **lastName: 'Albert',**  **age: 30,**  **joinedDate: 'February 15, 2011'**  **}**  **];**  **emp.sort((a,b)=>{**  **return a.age-b.age**  **})**  **console.log(emp)** |

**82.SORT ARRAY OF OBJECTS BASED ON DATE.**

|  |
| --- |
| **let emp = [**  **{**  **firstName: 'John',**  **lastName: 'Doe',**  **age: 27,**  **joinedDate: 'December 15, 2017'**  **},**  **{**  **firstName: 'Ana',**  **lastName: 'Rosy',**  **age: 25,**  **joinedDate: 'January 15, 2019'**  **},**  **{**  **firstName: 'Zion',**  **lastName: 'Albert',**  **age: 30,**  **joinedDate: 'February 15, 2011'**  **}**  **];**  **emp.sort((a, b) => {**  **let x = new Date(a.joinedDate);**  **let y= new Date(b.joinedDate);**  **return x - y;**  **});**  **console.log(emp)** |

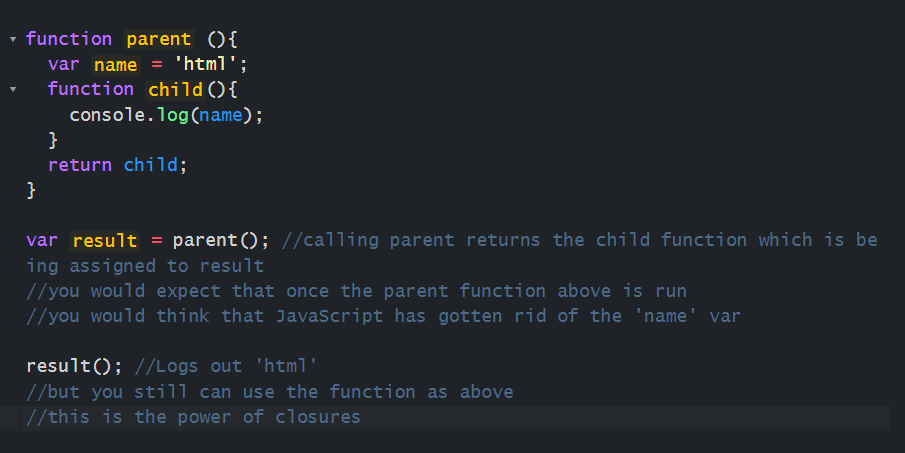
**83.LET AND CONST ALSO HOISTED?**

**LET AND CONST ALSO HOISTED BUT NOT INITILIZED WITH UNDEFINED.**

**84.CLOSURE**

**CLOSURE IS COMBINATION OF FUNCTION BUNDELED TOGETHER WITH BOUND TO THE LEXICAL SCOPE. IT WILL ACCESS OUTER FUNCTIONS VARIABLES AND AS WELL AS OUTER FUNCTION PARAMETERS.USING CLOSURES WE CAN IMPLEMENT DATA ENCAPSULATION,CURRRYING,WE CAN CREATE PRIVATE VARIABLES.**

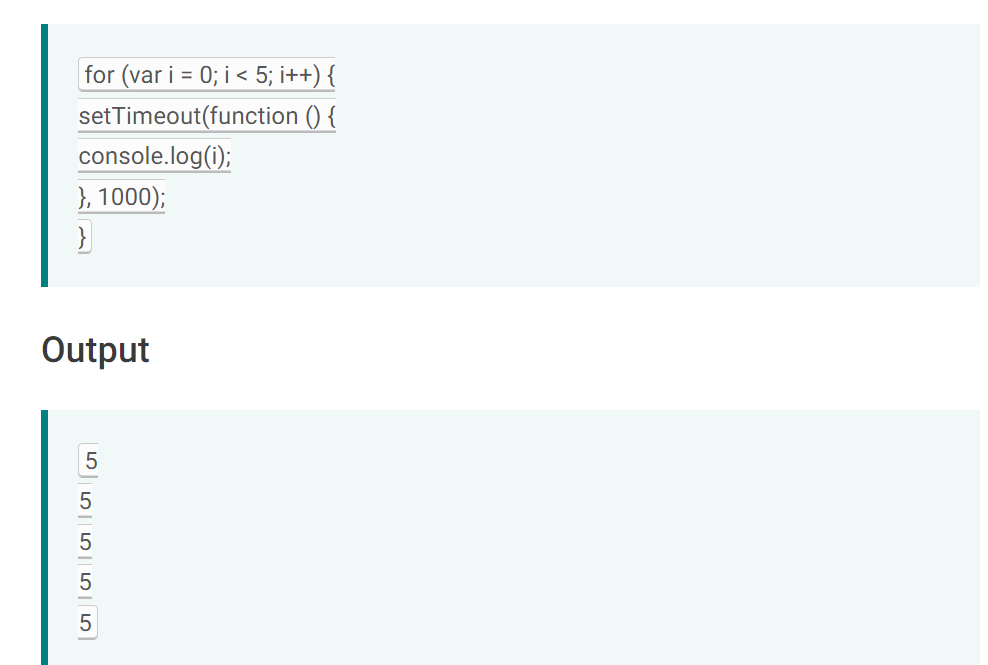
**85.CLOSURE EXAMPLE**

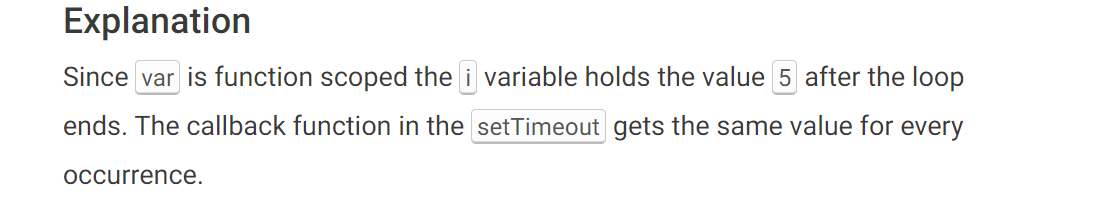
****

**86.CLOSURE WHEERE YOU ARE USED IN YOUR APPLICATION**

**IN MY MY PROJECT I HAVE IMPLEMENTED TYPEHEAD FEATURE,ADDITIONAL STDUY INFORMATION. THERE I HAVE USED**

**87.** **SNIPPET**

****

****

**88.HOW TO INCREASE PERFORMANCE JAVASCRIPT**

* **REMOVE UNUSED JAVASCRIPT**
* **PLACE SCRIPT TAG AT BOTTOM OF THE PAGE**
* **AVOID UNNECESSARY ACCESS DOM**
* **USE MINIFICATION**

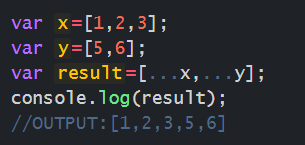
**89.CROSS SITE SCRIPTING(XSS)**

**CROSS SITE SCRIPTING IN WEB APPLICATION ALLOWS THIRD PARTY TO EXECUTE A SCRIPT IN THE USERS BROWSER BEHALF OF WEBAPPLICATION**

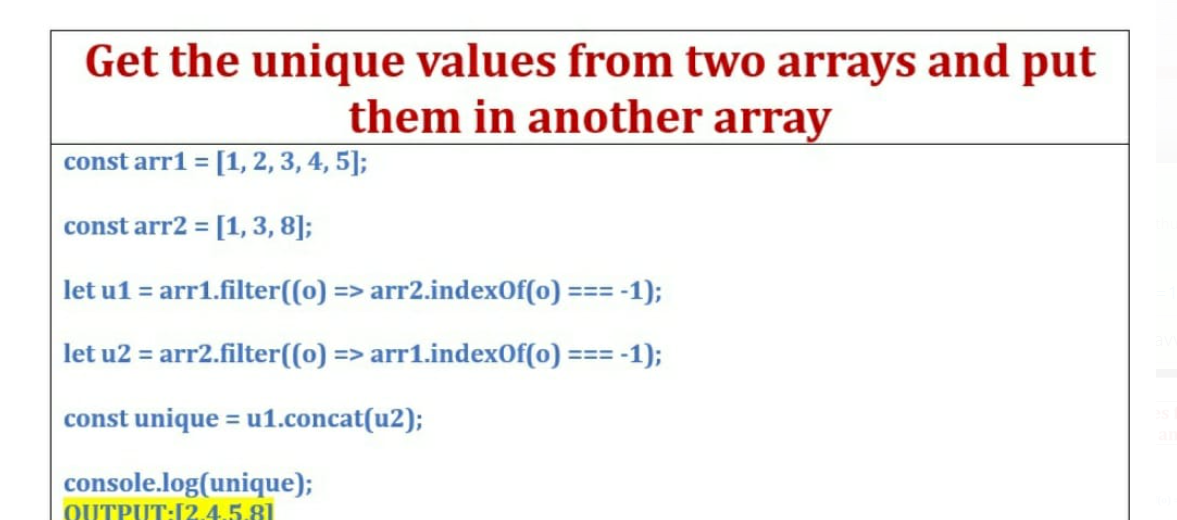
**90.HOW TO HANDLE SECURITY THREAD IN JAVASCRIPT**

* **MINIMIZE XSS**
* **CHECK THE REFERENCES EXTERNAL LIBRARIES**

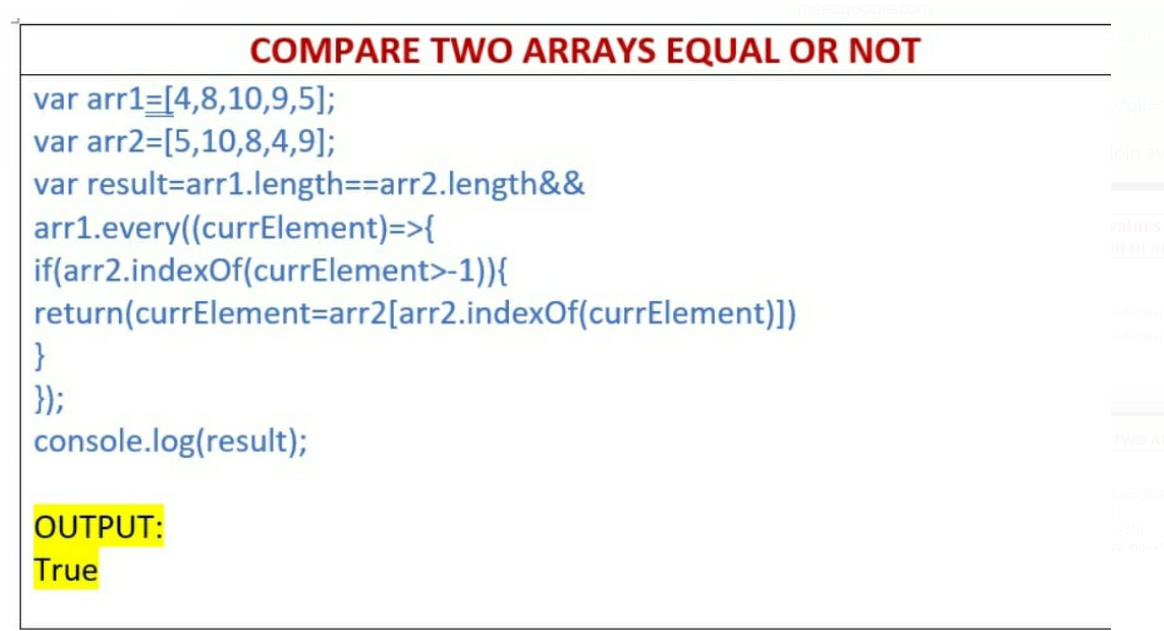
**91.HOW TO COMBINE TWO ARRAYS**



**92.**

****

**93.**

****

**94. HOW TO HANDLE EXCEPTIONS IN JAVASCRIPT?**

**WE CAN USE “TRY,CATCH,FINALLY”. IF ERROR IS PRESENT CATCH BLOCK WILL EXECUTE OTHERWISE TRY BLOCK WILL EXECUTE.**

**95.OBSERVABLES VS PROMISES**

|  |  |
| --- | --- |
| **PROMISES** | **OBSERVABLES** |
| **PROMISE EXECUTE IMMIDIATELY ON CREATION.** | **OBSERVABLES ARE DECLARATIVE** |
| **PROMISE DON’T HAVE PARAMETERS** | **OBSERVABLES PROVIDES OPERATORS LIKE MAP,FILTER,REDUCE.** |
| **PROMISE CANNOT BE CANCEL** | **OBSERVABLES CAN BE CANCEL** |

**96.HOW TO HANDLE BROWSER COMPATABILTY ISSUES?**

**IN CSS WE CAN USE PREFIXES AND CONDITIONAL STATEMENTS FOR CSS CODE**

**-MOZ- FOR “MOZILA”**

**-WEBKIT- FOR “CHROME AND SAFARI”**

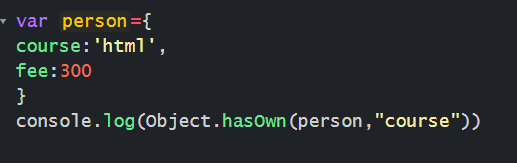
**-MS- FOR “INTERNET EXPROLER”**

**-O- FOR “OPERA”**

**WE CAN USE TRANSPILERS AND POLLYFILLS.**

**97.HOW TO CHECK PROPERTY EXIST OR NOT IN OBJECT.**

**WE CAN USE “Object.hasOwn” METHOD.**

****

**OUTPUT:TRUE**

**98.FUNCTION DECLARATION VS FUNCTION EXPRESSION**

|  |  |
| --- | --- |
| **FUNCTION DECLARATION** | **FUNCTION EXPRESSION** |
| **FUNCTION DECLARATION MUST HAVE FUNCTION NAME** | **FUNCTION EXPRESSION SIMILAR TO FUNCTION DECLARATION WITHOUT FUNCTION NAME.** |
| **FUNCTION DECLARATION DOES NOT REQUIRE VARIABLE ASSIGNMENT** | **FUNCTION EXPERSSION CAN BE STORED IN VARIABLE ASSIGNMENT** |
| **FUNCTION DECLARTION CAN BE ACCESSED BEFORE AND AFTER THE FUNCTION DEFINITION BECAUSE OF HOSTING** | **FUNCTION EXPRESSION CAN BE ACCESSED ONLY AFTER THE FUNCTION DECLARED. HERE HOSTING IT WON’T HAPPEN** |

**99.HOW JAVASCRIPT WORKS?**

* **EVERYTHING IN JAVASCRIPT HAPEENS INSIDE EXECUTION CONTEXT.**
* **EXECUTION CONTEXT IS LIKE CONTAINER AND IT HAS TWO COMPONENTS**

**1.MEMORY COMPONENT**

**2.CODE COMPONENT**

**MEMORY COMPONENT:**

**IN MEMORY COMPONENT ALL THE VARIBALES AND FUNCTIONS STORED AS KEY**

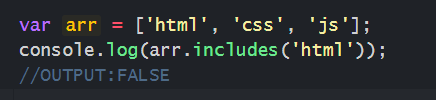
**VALUE PAIRS. MEMORY COMPONENT ALSO KNOWN AS VARIABLE ENVIRONMENT.**

**CODE COMPONENT: IN CODE COMPONENT CODE IS EXECUTED ONE LINE AT A TIME.IT IS ALSO KNOWN AS THREAD OF EXECUTION.**

* **JAVASCRIPT IS A SYNCHRONOUS SINGLE THREADED LANGUAGE. SINGLETHREAD MEANS JAVASCRIPT ONLY EXECUTE ONE LINE AT A TIME.SYNCHRNOUS SINGLE THREAD MEANS JAVASCRIPT ONLY EXECUTE ONE COMMAND AT A TIME IN SPECIFIC ORDER.**

**100.HOW TO CHECK ELEMENT IS EXIST IN ARRAY OR NOT.**

**WE CAN USE “INCLUDES” METHOD.**

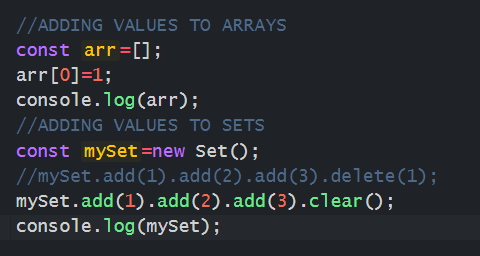
****

**101.SETS**

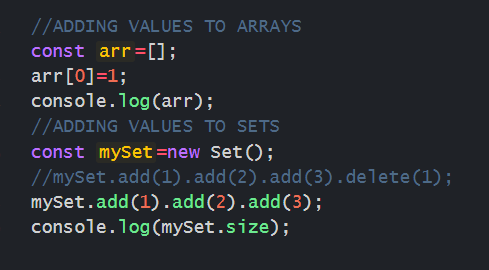
* **SETS ARE ACTUALLY REFERENCE DATA TYPE**
* **SETS ARE LIKE ARRAY THEY ARE REFERENCE DATA TYPE**
* **THEY ALLOW TO STORE UNIQUE VALUES OF ANY DATA TYPES MEANS WE CANNOT HAVE DUPLICATE VALUES INSIDE SET.**
* **ARRAY LITERAL: const arr=[];**

**ARRAY CONSTRUCTOR: const myArray=new Array();**

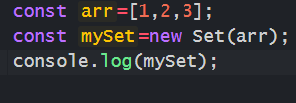
* **SETS WE HAVE TO CONSTRUCTOR IT DOES NOT HAVE LITERAL SYNTAX.**
* **YOU CANNOT ACCESS SETS LIKE YOU WOULD ACCESS ARRAYS. I HAVE TO USE PROTOTYPE METHOD**
* **JUST LIKE ARRAY HAVE “LENGTH” SETS HAS “SIZE”**

**EXAMPLE:  
========  
  
**

**EXAMPLE:  
========**

****

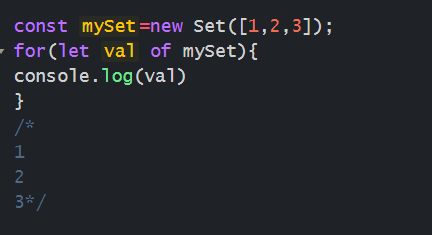
* **NOW I DON’T ALWAYS HAVE TO ADD LIKE THIS I CAN ACTUALLY CREATE AN ARRAY AND CONVERT THAT ARRAY TO SET.**

****

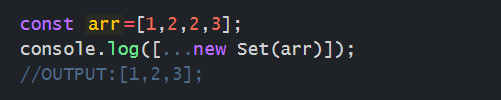
**OR**

****

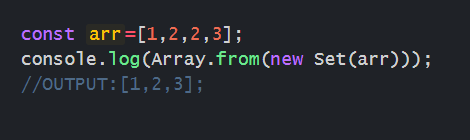
* **SET IS ONLY TAKES ARGUMENT SOMETHING THAT IS ITERABLE SO ARRAYS ARE ITERABLE IT WILL TAKE ARRAYS. IT WONT TAKE JAVACRIPT WHICH IS NOT ITERABLE.**
* **SETS ARE ITERABLE LIKE ARRAYS.**
* **IF I LOOK PROTOTYPE WE WOULD SEE “Symbol.Iterator()”.**

****

* **SETS ARE NOT ARE ARRAYS AND ALSO NOT KEY-VALUE PAIRES LIKE OBJECTS.**
* **WE CAN ITERATE THEM LIKE ARRAYS BECAUSE IT HAS ENTRIES LIKE ARRAYS BUT THEY ARE NOT ARRAYS**
* **ARRAYS HAVE LOT OF METHODS LIKE MAP,REDUCER,FILTER ETC. SET DOES NOT HAVE.**
* **SETS CAN BE USEFUL WHEN YOU WANT TO HAVE TWO SETS YOU WANT TO INTERSECT AND YOU WANT TO MAKE UNION FIND DIFFERENCES THAT’S WHY VERY USEFUL.**
* **SETS WILL REMOVE DUPLICATE VALUES FROM ARRAYS.**

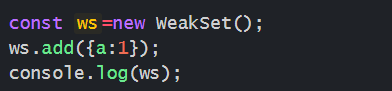
****

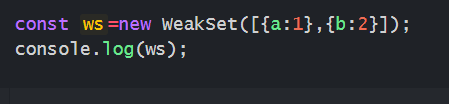
**OR**

****

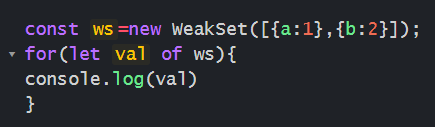
**102.WEAK SETS**

* **LIKE SETS WE CANNOT ADD PRIMITIVE VALUES YOU CAN ONLY ADD OBJECTS.**

****

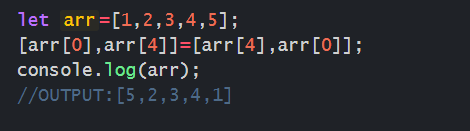
****

* **WEAKSETS ARE NOT ITERABLE OBJECTS.**

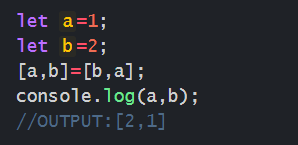
****

**ERROR: “WS” IS NOT ITERABLE**

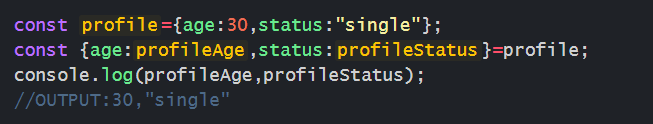
**102.SNIPPET**

**]**

**103.SWAPING VARIABLES**

****

**104.**

****

**105.HOW TO CONVERT BOOLLEAN TO STRING?  
USING “String” METHOD.  
var x=String(true);**

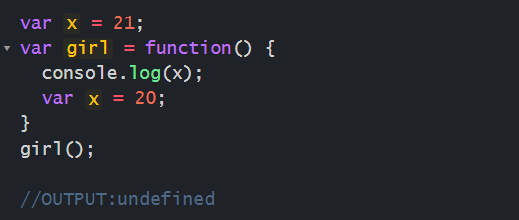
**106.401 STATUS CODE**

**UNAUTHORIZED ERROR. THIS IS CLIENT SIDE ERROR. IF USER CREDENTIALS ARE INCORRECT AND SERVER DOESN’T WHO IS USER IS THAT TIME WE WILL GET THIS ERROR.**

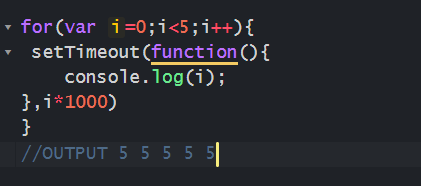
**107.403 STATUS CODE**

**403 IS A FORBIDDEN ERROR. THIS IS CLIENT SIDE ERROR. WHEN WEBPAGE SERVER DENIES ACCESS TO A WEBPAGE**

**108.SNIPPET**

****

**109.**

****

**110.console.log([1,2,3]+[4,5,6]);**

**‘1,2,34,5,6’**

**111.SETTIMEOUT VS SETINTERVAL?**

**setTimeout**

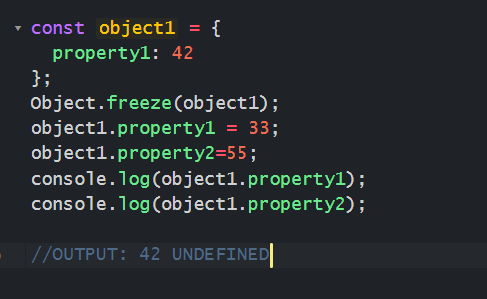
**EXECUTES A FUNCTION,AFTER WAITING SPECIFIED NUMBER OF MILLISECONDS.**

**setInterval**

**SAME AS SETTIMEOUT, BUT REPEATS THE EXECUTION OF THE FUNCTION CONTINUOSLY.**

**112.** **THERE IS AN OBJECT WE CAN MODIFY EXIST AND WE CANNOT ADD NEW VALUE**

**WE CAN USE OBJECT.FREEZE**

****

**113.indexOf**

**THE “indexOf” METHOD RETURNS THE POSITION OF THE FIRST OCCURRENCE OF VALUE IN STRING.**

**IT RETURS -1 IF VALUE IS NOT FOUND.**

**IT IS CASE SENSITIVE**

**114.lastIndexOf**

**THE “lastIndexOf” METHOD RETURNS THE POSITION OF THE LAST OCCURRENCE OF VALUE IN STRING.**

**IT RETURS -1 IF VALUE IS NOT FOUND.**

**IT IS CASE SENSITIVE**

**115.FILTER**

* **FILTER METHOD CREATES NEW ARRAY FROM GIVEN ARRAY WHICH SATISFY CONDITION.**
* **FILTER METHOD DOES NOT EXECUTE FOR EMPTY ELEMENTS**
* **FILTER METHOD DOES NOT CHANGE ORIGINAL ARRAY**
* **FILTER METHOD ACCEPTS 5 PARAMETERS**

**CALLBACK**

**CURRENT VALUE**

**INDEX(OPTIONAL)**

**ARRAY(OPTIONAL)**

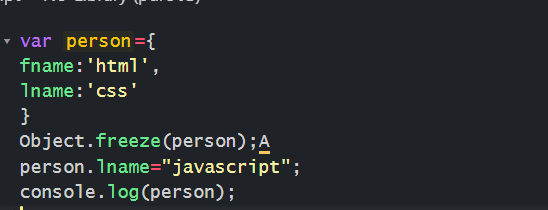
**THIS(OPTIONAL)**

**116.IF FILTER METHOD NOT SATISFIED WHAT IT WILL RETURN**

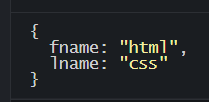
**[]**

**117.HOW TO PREVENT ADD NEW PROPERTIES FROM OBJECT?**

**WE CAN USE “Object.freeze()” METHOD.**

****

**OUTPUT:**

****

**118.WHAT SITUATION YOU WILL USE CLOSURES.**

* **CLOSURES CAN BE USED TO CREATE PRIVATE VARIABLES**
* **IT CAN BE USED TO CREATE CALLBACK FUNCTION**
* **IT CAN BE USED EVEN OUTER FUNCTION FINISHED EXECUTING.**

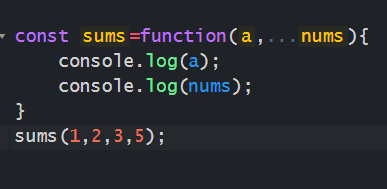
**119.WHICH SITUATION YOU WILL USE ARROW FUNCTIONS**

**TO DEFINE CALLBACK FUNCTIONS, WHEN YOU NEED TO USE ‘THIS’ KEYWORD OR ARGUMENT OBJECT IN CODE.**

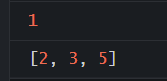
**120.SNIPPET**

****

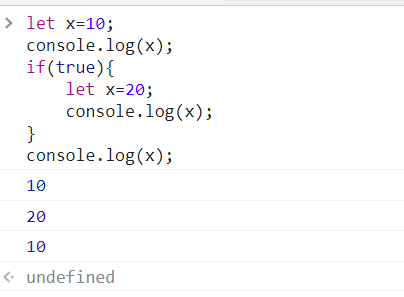
**121.SNIPPET**

****

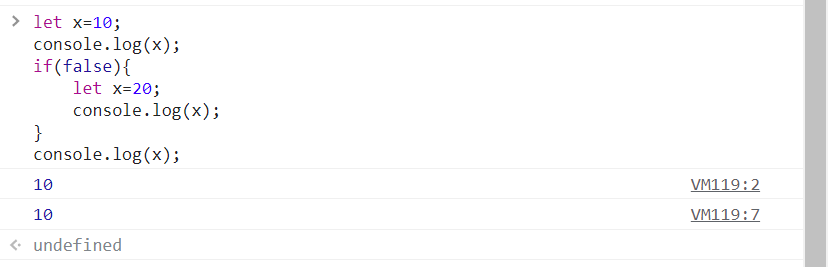
**OUTPUT**

****

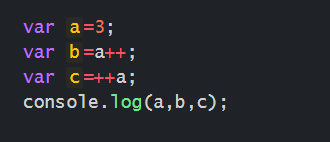
**122.SNIPPET**

****

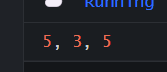
**123.SNIPPET**

****

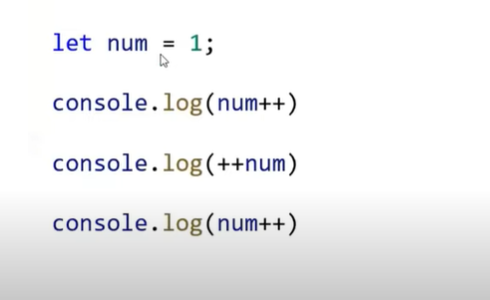
**124.SNIPPET**

****

**OUTPUT**

****

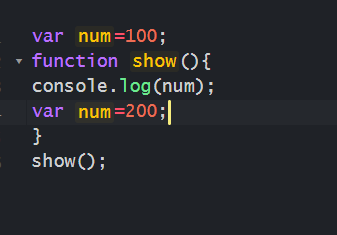
**125.SNIPPET**

****

**OUTPUT:**

**1,3,3**

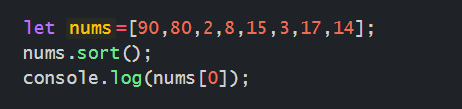
**126.SNIPPET**

****

**OUTPUT:**

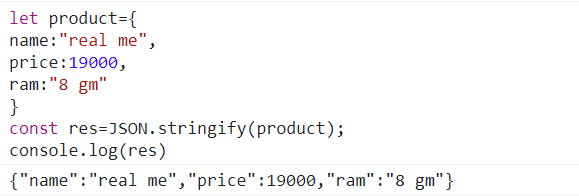
**UNDEFINED**

**127.SNIPPET**

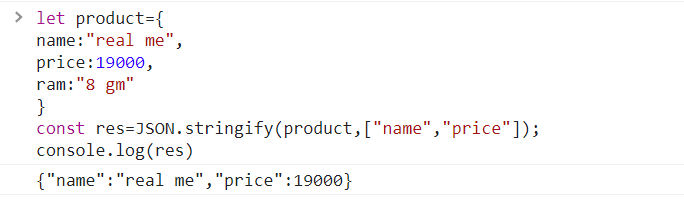
****

**OUTPUT: 14**

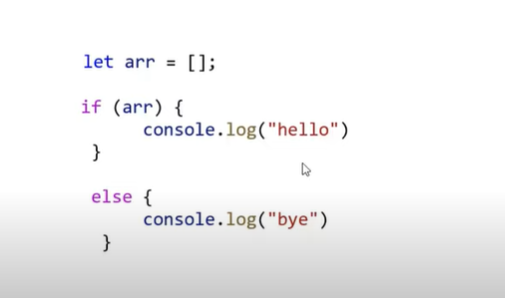
**128.SNIPPET**

****

**129.SNIPPET**

****

**130.SNIPPET**

****

**OUTPUT: HELLO**

**131.HOW DO YOU MAKE OBJECT IMMUTABLE?**

**WE CAN USE Object.freeze() METHOD.**

**132.WHAT IS DEBOUNCING**

**DEBOUNCING TECHNIQUE DELAYS THE EXECUTION OF FUNCTION UNTIL CERTAIN AMOUNT OF TIME HAS PASSED.**

**WHEN USER TYPING INPUT FIELD,INSTEAD OF EXECUTION THE FUNCTION EVERY KEYSTROKE.FUNCTION IS EXECUTED AFTER SOME TIME SINCE LAST KEYSTROKE.**

**133.THROTLING**

**DEBOUNCING SAME AS THROTLING.DIFFRENCE IS WHEN USER IS SCROLLING THROUGH PAGE INSTEAD OF EXECUTING THE FUNCTION EVERY SCROLL EVENT. THE FUNCTION IS ONLY EXECUTED CERTAIN NUMBER OF TIMES PER SECOND. DEBOUNCING AND THROTLING IMPROVES THE PERFORMANCE OF THE APPLICATION.**

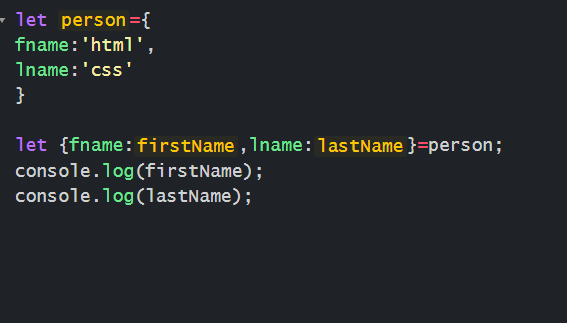
**134.eventPreventDefault()**

**“EVENTPREVENTDEFAULT()” METHOD IS USED PREVENT DEFAULT ACTION OF EVENTS LIKE LINK CLICKS,BUTTON CLICKS AND KEYBOARD EVENTS.**

**135.event.stopPropagation()**

**event.stopPropagation() METHOD IS USED TO STOP THE PROPAGATION OF EVENT THROUGH THE DOM.**

**136.HOW TO RENAME OBJECT PROPERTIES WITH DESTRUCTURING**

****