ASSIGNMENT 1

Q1. Why dot use with Value Type?

- provide access to an object's properties by using the dot notation

Dot notation is used when object name is valid. Example

object.propertyname here propertyname must be valid

The same work can be done by bracket notation. Object can be represented by bracket notation too. In the object[expression] syntax, the expression should evaluate to a string that represents the property's name. Bracket Notation is useful when you want to dynamically access the property.

Q-2,4) What is null value and where we use it? Hint: typeof

Undefined simply means that when we are taking any variable and not assigning any value to it. By default every non assigned var is undefined.

Example: var name;

Undefined

Typeof name;

undefined

-a **null** value represents a reference that points, generally to invalid object or address. Generally represents assignment value.

Example : var object={name}

Object .name;

Value will be name

object =null;

Value will be null

Q 3 – typeof

This operator returns a string representing type of the operand value

console .log(typeof 42);

// Expected output: "number"

console .log(typeof 'blubber');

// Expected output: "string"

console .log(typeof true);

// Expected output: "boolean"

console .log(typeof undeclaredVariable);

// Expected output: "undefined"

Q5 == vs ===

* [===](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Strict_equality) — strict equality (triple equals)
* [==](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Equality) — loose equality (double equals)

* Double equals (==) will perform a type conversion when comparing two things, and will handle NAN.
* Triple equals (===) will do the same comparison as double equals (including the special handling for NAN
* Strict equality is almost always the correct comparison operation to use. For all values except numbers, it uses the obvious semantics: a value is only equal to itself. This is useful in representing certain mathematical solutions, but as most situations don't care about the difference between +0 and -0, strict equality treats them as the same value. The second is that floating point includes the concept of a not-a-number value, NAN to represent the solution to certain ill-defined mathematical problems: negative infinity added to positive infinity, for example. Strict equality treats NAN as unequal to every other value — including itself. (The only case in which (x === x) is true is when x is NAN.)
* for performing loose equality using == is as follows:
* If the operands have the same type ==will return true else false.
* If one of the operands is null or undefined, the other must also be null or undefined to return true. Otherwise return false.

Q 6 Prime Number Compute Function

function primeno(a) {

var flag = false;

for (let i=2;i<=a;i++){

if(a%i==0){

flag =true;

break;

}

}

if(flag == true){

console.log("number is not prime");

}

else{

console.log("prime number");

}}

Q7. ArmStrong Number Compute Function

let sum = 0;

const number = prompt('Enter a three-digit positive integer: ');

let temp = number;

while (temp > 0) {

let remainder = temp % 10;

sum += remainder \* remainder \* remainder;

temp = parseInt(temp / 10);

}

if (sum == number) {

console.log(`${number} is an Armstrong number`);

}

else {

console.log(`${number} is not an Armstrong number.`);

}

Q8 add() - 0

add (10) - 10

add (10,20) - 30

add (10,”20”) - 30

add (10,”ten”) - 10

add (10, NAN) - 10

add (10, 20, 30 ,40, 50)- 150

function add(){

var sum=0

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

if((typeof(arguments[i]))==’string’){

arguments[i]=parseInt(arguments[i]);

if(isNaN(arguments[i])){

sum=sum+0;

}

Else{

Sum=sum+arguments[i];}}

Else if ((typeof(arguments[i]))==’number’){

Sum+=arguments[i];

}

else if(arguments[i] instanceof array){

var sum1=0;

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

arguments[i][j]=parseInt(arguments[i][j];

if(isNaN(arguments[i][j])){

sum1=sum1+0;

}}

Console.log(sum);

}

Add(1,2,3,4,”why”,[1])