Block of code that performs a specŤfŤc task, can be Ťnvoked whenever needed

FunctŤon DefŤnŤtŤon

FunctŤon Call

**function** functionName**( ) {**

//do some work

functionName**( );**

**}**

**function** functionName**( param1, param2 ...) {**

//do some work

**}**

# Arrow FunctŤons

Compact way of wrŤtŤng a functŤon

const functionName **= ( param1, param2 ...) => {**

//do some work

**}**

const sum = ( a, b ) => {

return a + b;

}

# Let‘s PractŤce

Qs. Create a functŤon usŤng the “functŤon” keyword that takes a StrŤng as an argument & returns the number of vowels Ťn the strŤng.

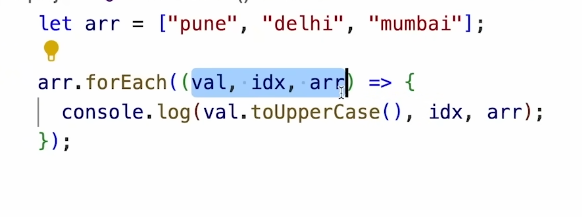
Qs. Create an arrow functŤon to perform the same task.

# forEach Loop Ťn Arrays

arr.**forEach**( callBackFunction ) call back function should not have () while calling in the higer function

CallbackFunctŤon : Here, Ťt Ťs a functŤon to execute for each element Ťn the array

\*A callback Ťs a functŤon passed as an argument to another functŤon.

val, ind, arr 

arr.forEach( ( val ) => {

console.log(val);

} )

# Let‘s PractŤce

Qs. For a gŤven array of numbers, prŤnt the square of each value usŤng the forEach loop.

Map

Creates a new array wŤth the results of some operatŤon. The value Ťts callback returns are used to form new array

arr.**map(** callbackFnx( value, index, array ) **)**

let newArr = arr.map( ( val ) => {

return val \* 2;

} )

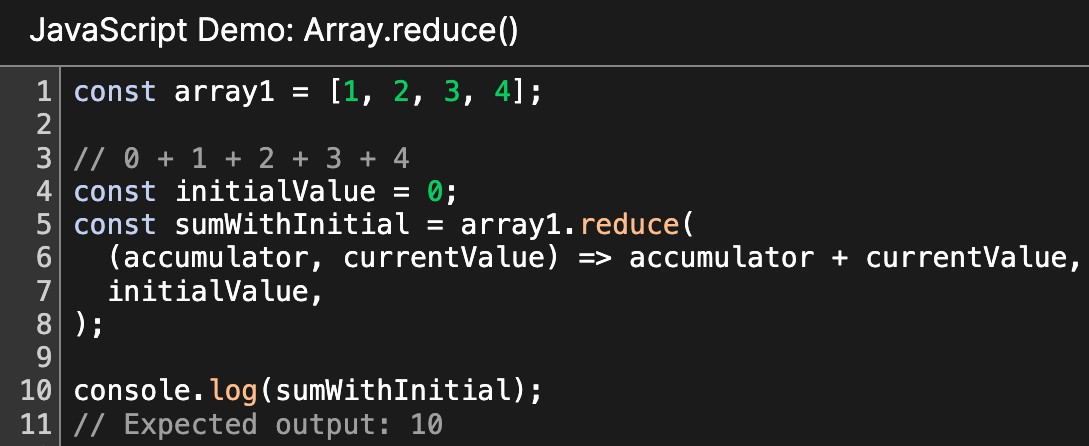
FŤlter

Creates a new array of elements that gŤve true for a condŤtŤon/fŤlter. Eg: all even elements

let newArr = arr.filter( ( ( val ) => {

return val % 2 === 0;

} )



Reduce

Performs some operatŤons & reduces the array to a sŤngle value. It returns that sŤngle value.

# Let‘s PractŤce

Qs. We are gŤven array of marks of students. FŤlter our of the marks of students that scored 90+.

Qs. Take a number n as Ťnput from user. Create an array of numbers from 1 to n. Use the reduce method to calculate sum of all numbers Ťn the array.

Use the reduce method to calculate product of all numbers Ťn the array.