ARRAY REDUCE( )

The Javascript arr.reduce() method in JavaScript is used to reduce the array to a single value and executes a provided function for each value of the array (from left to right) and the return value of the function is stored in an accumulator.

Syntax:

array.reduce(function(total, currentValue, currentIndex, arr),

initialValue)

Parameters: This method accepts five parameters as mentioned above and described below:

function(total, currentValue, index, arr): It is the required parameter and is used to run for each element of the array. It contains four parameters which are listed below:

total: It is a required parameter and used to specify the initialValue or the previously returned value of the function.

currentValue: It is a required parameter and is used to specify the value of the current element.

currentIndex: It is an optional parameter and is used to specify the array index of the current element.

arr: It is an optional parameter and is used to specify the array object the current element belongs to.

initialValue: It is an optional parameter and is used to specify the value to be passed to the function as the initial value.

This example uses reduce() method to return the sum of all array elements.

let arr = [10, 20, 30, 40, 50, 60];

```
function sumofArray(sum, num) {
   return sum + num;
}
function demo(item) {
   console.log(arr.reduce(sumofArray));
}
demo()
```

JavaScript setTimeout() method

The **setTimeout()** method in JavaScript is used to execute a function after waiting for the specified time interval. This method returns a numeric value that represents the ID value of the timer.

Syntaxt

1. setTimeout(function, milliseconds);

ex
```
function demo( ) {
  console.log("hello")
}
setTimeout(demo,2000)
```

JavaScript setInterval() method

The setInterval() method in JavaScript is used to repeat a specified function at every given time-interval. It evaluates an expression or calls a function at given intervals. This method continues the calling of function until the window is closed or the clearInterval() method is called

How to stop the execution?

We can use the **clearInterval()** method to stop the execution of the function specified in **setInterval()** method. The value returned by the **setInterval()** method can be used as the argument of **clearInterval()** method to cancel the timeout.

Syntaxt

1. setInterval(function, milliseconds);

```
let ele=document.getElementById("print")
        // let num=0;
        // let timer;
        //   function setTimeInterval() {
        //       ele.innerHTML="loading...."
        //       timer=setInterval((welcome)=>{
        //           ele.innerHTML=`i m guessing your lucky number  is it
${num}`
        //           ++num;
        //       },1000)
        //   }
        //   function stopTimer() {
        //     clearInterval(timer)
        //   }
```

ES6 Rest Parameter

The rest parameter is introduced in ECMAScript 2015 or ES6, which improves the ability to handle parameters. The rest parameter allows us to represent an indefinite number of arguments as an array. By using the rest parameter, a function can be called with any number of arguments.

Before ES6, the **arguments** object of the function was used. The **arguments** object is not an instance of the Array type. Therefore, we can't use the **filter()** method directly.

The rest parameter is prefixed with three dots (...). Although the syntax of the rest parameter is similar to the spread operator, it is entirely opposite from the spread operator. The rest parameter has to be the last argument because it is used to collect all of the remaining elements into an array.

Syntax …variable

```
function show(...args) {
  let sum = 0;
  for (let i of args) {
     sum += i;
  }
  console.log("Sum = "+sum);
}

show(10, 20, 30);
```