

# SISTEMAS NUMÉRICOS: Introducción a la Informática

JAIDER ALBERTO RENDON MORENO  
OCTUBRE DE 2020;JAIDER ALBERTO RENDÓN MORENO



# 1 CONTENIDO

---

1	CONTENIDO.....	1
2	PRESENTACIÓN.....	2
3	CONVERSIÓN BASADA EN DIVISIONES SUCESIVAS.....	3
4	CONVERSIÓN EXTENDIDA A BASE 16 CON SWITCH.....	6
5	CONVERSIÓN: 0, 1 A PALABRAS: CERO, UNO.....	9
6	CONTAR NÚMERO DE UNOS EN UN BINARIO.....	11
7	ÍNDICE DEL NÚMERO BINARIO.....	15
8	ÍNDICE DEL NÚMERO BINARIO INVERSO.....	16
9	CONVERSIÓN DE BINARIO A DECIMAL.....	17
10	RECURSIÓN.....	18
11	MULTIPLICACIÓN CON RECURSIÓN.....	20
12	ITERACIÓN.....	21
13	CONCLUSIONES.....	22
14	REFERENCIAS.....	23



## 2 PRESENTACIÓN

---

La presente monografía describe la implementación de un conjunto de programas que le dan soporte a la teoría numérica básica de la materia INTRODUCCIÓN A LA INFORMÁTICA.

En los siguientes párrafos se presenta una descripción básica del significado de lo que es un sistema numérico, especialmente el sistema en base 2.

Cabe resaltar que utilizaremos el medio de desarrollo de entornos web como lo es el HTML y el JavaScript para llevar a cabo los algoritmos.

AUTOR: Jaider Alberto Rendón Moreno

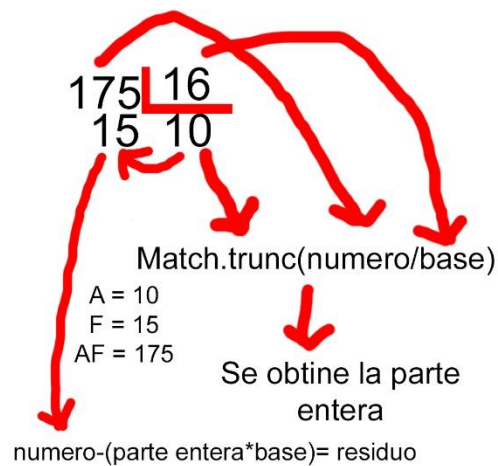
1006149333

Jaider.rendon@utp.edu.co

<https://github.com/VOIDX66/Development-of-the-void>

### 3 CONVERSIÓN BASADA EN DIVISIONES SUCESIVAS

A continuación se presenta el algoritmo básico para la conversión numérica basada en divisiones sucesivas.



Como se ve en el diagrama, la conversión se realiza dividiendo el número a convertir entre la base seleccionada.

El resultado se obtiene con base en los residuos de las divisiones.

El proceso finaliza cuando se obtiene cero en el resultado de las divisiones.

A continuación se presentan las imágenes de los códigos requeridos, para implementar el proceso mostrado en JavaScript. Cada imagen presenta una función distinta, o la ejecución final del programa. Se debe escribir en un solo archivo el código mostrado, y se sugiere un entorno como repl.it.

```
function texto( cadena, num_saltos = 0 ) {
    document.write( cadena );
    var i = 0;
    while (i < num_saltos ) {
        document.write( "<br />" );
        i = i + 1;
    }
}
```

```
function conversion( numero, base ) {
  var division, resto;
  var result = "";
  var control = 0;
  var bandera = 0;
  while ( bandera == 0 ) {
    ///////////////////////////////////
    division = Math.trunc( numero / base );
    resto = numero - division * base;
    result = resto.toString() + result;
    numero = division;
    if (numero <= 0) {
      bandera = 1;
    }
    control = control + 1;
    if ( control > 1000 ) {
      bandera = 1;
    }
  }
  return result;
}
```

```
texto( "PROGRAMA DE CONVERSIÓN NUMÉRICA", 1);
texto( "Octubre 13 de 2020");
texto( "", 2);

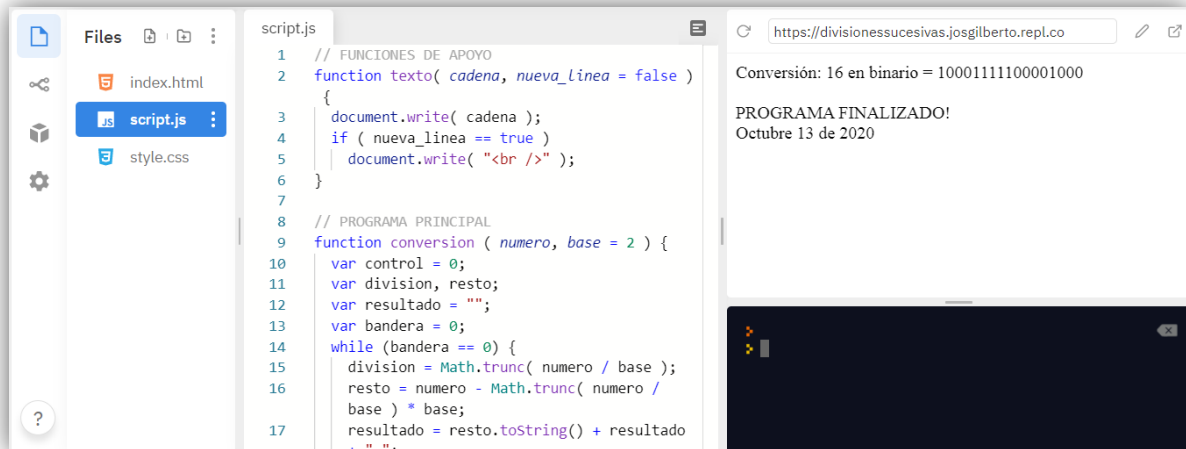
var n = 83; // El número a convertir
var b = 16; // La base de conversión

var resultado;
resultado = conversion( n, b );

texto( "Número: " + n, 1);
texto( "Base:   " + b, 1);
texto( "Resp:   " + resultado, 1);

</script>
```

A continuación se muestra el programa en el entorno repl.it, con los datos de ejecución del programa.



```

1 // FUNCIONES DE APOYO
2 function texto( cadena, nueva_linea = false )
3 {
4   document.write( cadena );
5   if ( nueva_linea == true )
6     document.write( "<br />" );
7 }
8
9 // PROGRAMA PRINCIPAL
10 function conversion ( numero, base = 2 ) {
11   var control = 0;
12   var division, resto;
13   var resultado = "";
14   var bandera = 0;
15   while (bandera == 0) {
16     division = Math.trunc( numero / base );
17     resto = numero - Math.trunc( numero /

```

Conversion: 16 en binario = 10001111100001000

PROGRAMA FINALIZADO!  
Octubre 13 de 2020

Por medio de la función `Math.trunc()` se obtiene siempre un resultado entero por lo tanto no hay problemas con números decimales y sabremos con exactitud el residuo de la operación, este algoritmo se lleva a cabo por medio de diferentes `while` y `if` con los cuales controlaremos los valores de las divisiones sucesivas.

## 4 CONVERSIÓN EXTENDIDA A BASE 16 CON SWITCH

A continuación presentamos el programa de conversión extendida, la cual se encarga de dar tratamiento a los números en base 16.

Para la base 16 tendremos algo en particular ya que desde el número 10 hasta el 15 se representara por letras tal que.

El código es el siguiente:

```
// PROGRAMA PRINCIPAL
function conversion ( numero, base ) {
    var control = 0;
    var division, resto;
    var resultado = "";
    var bandera = 0;
    while (bandera == 0) {
        division = Math.trunc( numero / base );
        resto = numero - Math.trunc( numero / base ) * base;

        if (base == 16 && resto > 9 ) {
            var aux = "";
            switch (resto) {
                case 10:
                    aux = "A";
                    break;
                case 11:
                    aux = "B";
                    break;
                case 12:
                    aux = "C";
                    break;
                case 13:
                    aux = "D";
                    break;
                case 14:
                    aux = "E";
                    break;
            }
        }
        resultado = aux + resultado;
        numero = division;
        if (numero == 0) {
            bandera = 1;
        }
    }
    return resultado;
}
```

DECIMAL		HEXADECIMAL
1	=	1
2	=	2
3	=	3
4	=	4
5	=	5
6	=	6
7	=	7
8	=	8
9	=	9
10	=	A
11	=	B
12	=	C
13	=	D
14	=	E
15	=	F

```

        aux = "E";
        break;
    case 15:
        aux = "F";
        break;
    default:
        aux = "";
        break;
    }
    resultado = aux + resultado + " ";
}
else {
    resultado = resto.toString() + resultado + " ";
}

numero = division;

if ( division < 1 )
    bandera = 1;

control = control + 1;
if (control > 1000) {
    texto( "ERROR. Se superó el número de 1000 iteraciones", 1 );
    bandera = 1;
}
}

```

```

    return resultado;
}

// EJECUCIÓN DEL PROGRAMA
var n = 175;
var b = 16;

var resp = conversion( n, b );

texto( "Número: " + n, 1);
texto( "Conversión a base: " + b, 1);
texto( "Resultado: " + resp, 2);

// PROGRAMA TERMINADO
texto("PROGRAMA FINALIZADO!", 1 );
texto("Octubre 22 de 2020");

```





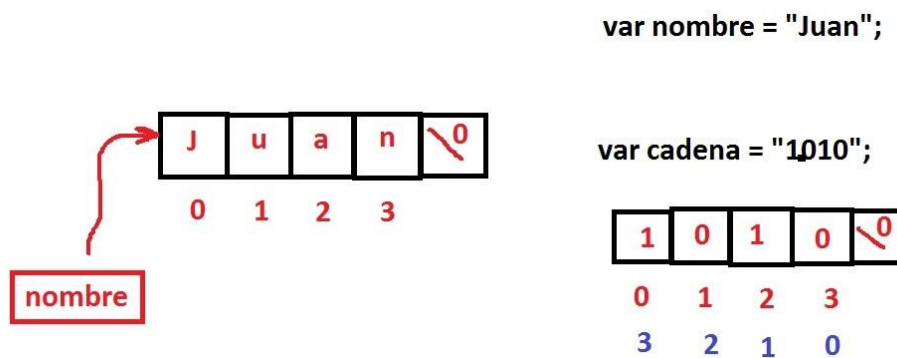
Entorno de Repl.it:

Basándonos en la lógica del programa anterior utilizaremos el `Math.trunc()` para obtener la parte entera y poder tener un residuo, al igual que la utilidad de la variable bandera, pero ahora se suma una condición, este es que mientras la base se igual a 16 y (AND operador lógico representado con `&&` en el código) resto mayor que 9 se ejecuta en switch en el cual todo número mayor de nueve hasta 15 tendrá una letra asignada ya que así funciona en el Hexadecimal.

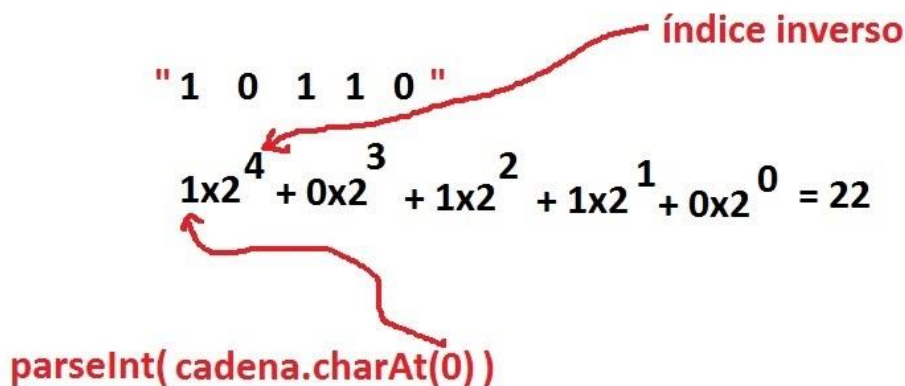
## 5 CONVERSIÓN: 0, 1 A PALABRAS: CERO, UNO

Vamos a presentar el programa que convierte los dígitos binarios 0, 1 a las palabras Cero, Uno.

Algunos elementos teóricos. En primer lugar, observamos cómo se almacena una cadena en la memoria:



En la siguiente gráfica, presentamos el índice inverso:



La función `parseInt()` recibe como dato inicial una cadena para convertirla en un dato numérico.

A continuación, presentamos el código fuente, seccionado por partes:

```
3 // PROGRAMA - 01
4 // -----
5 // Programa que permite leer en una cadena de texto, un número binario
6 // Los ceros y los unos contenidos en la cadena de texto
7 // Son convertidos a las palabras: Cero y Uno, respectivamente
8
9 var cadena = prompt("Introduzca un número binario");
10 var salida = "";
11
12 var i = 0;
13 while ( i < cadena.length ) {
14
15     if (cadena.charAt(i) === "0") {
16         salida = salida + "Cero ";
17     }
18     if (cadena.charAt(i) === "1") {
19         salida = salida + "Uno ";
20     }
21
22     i = i + 1;
23 }
```

En estas líneas se comenta el programa, se crean algunas variables, se pregunta por una cadena binaria y se procede luego a realizar la conversión, dígito a dígito.

Ejemplo de la ejecución:

script.js

```

1  var cadena = prompt("Introduzca un número binario");
2
3  var salida = "";
4
5  var i = 0;
6
7  while(i < cadena.length ){
8
9      if(cadena.charAt(i) === "0"){
10         salida = salida + "Cero";
11     }
12
13     if(cadena.charAt(i) === "1"){
14         salida = salida + "Uno";
15     }
16
17     i = i + 1;
18 }
19
20 document.write("Cadena = " + cadena + "<br></br>");
21
22 document.write("Salida = " + salida + "<br></br>");
23
24 document.write("FINALIZADO");

```

La variable salida esta inicialmente en blanco porque ahí es donde se pondrán las palabras "Cero" y "Uno" concatenadas dependiendo del binario ingresado.

## 6 CONTAR NÚMERO DE UNOS EN UN BINARIO

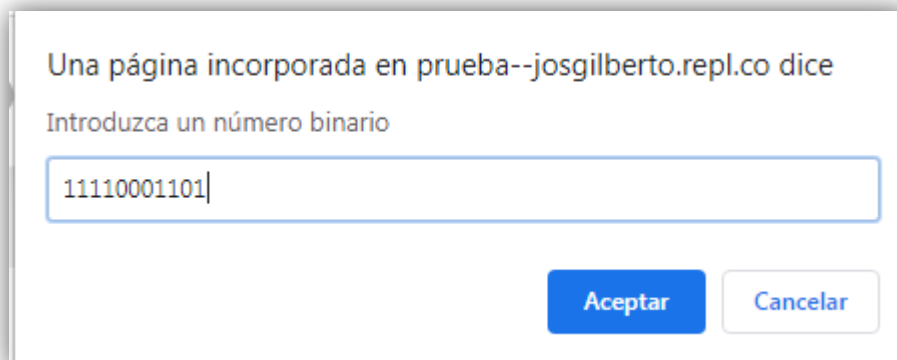
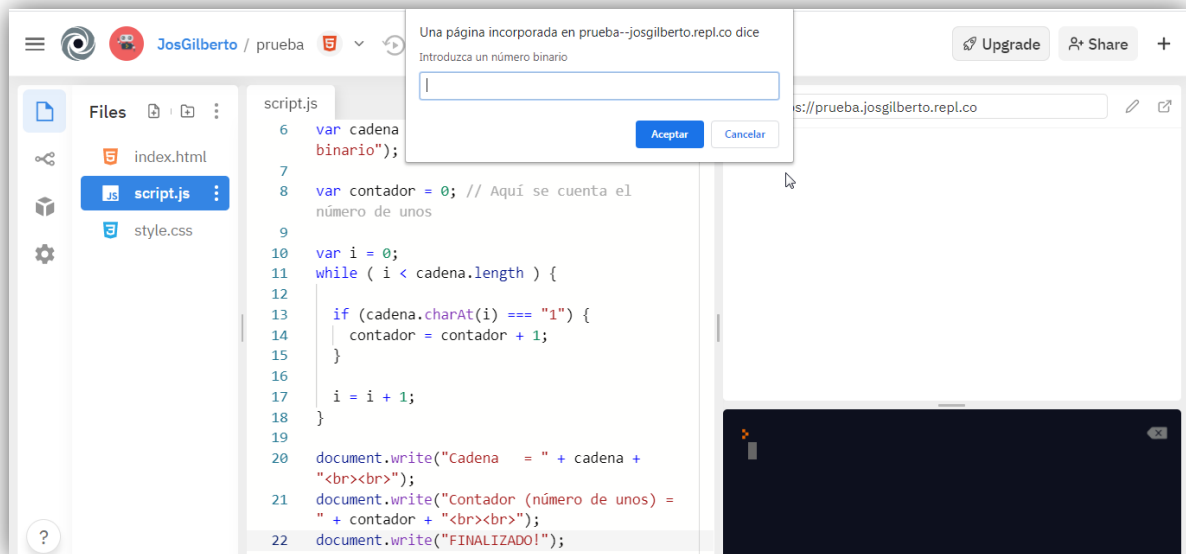
El número de unos se calcula recorriendo la cadena de izquierda a derecha, evaluando cada una de las posiciones en la cadena, y cada vez que el contenido de dicha posición sea uno, sumaremos 1 a un contador preparado para ello.

Seguidamente presentamos el código fuente.

```
1 <script>
2 // PROGRAMA - 02
3 // -----
4 // Programa que permite leer en una cadena de texto, un número binario
5 // El programa cuenta el número de unos
6
7 var cadena = prompt("Introduzca un número binario");
8
9 var contador = 0; // Aquí se cuenta el número de unos
10
11 var i = 0;
12 while ( i < cadena.length ) {
13
14     if (cadena.charAt(i) === "1") {
15         contador = contador + 1;
16     }
17
18     i = i + 1;
19 }
20
21 document.write("Cadena    = " + cadena + "<br><br>");
22 document.write("Contador (número de unos) = " + contador + "<br><br>");
23 document.write("FINALIZADO!");
24 </script>
```

Mientras que la variable *i* que tiene valor inicial 0 sea menor que la longitud de la cadena en la cual se introdujo un número binario se ejecutará una condicional en la cual cada que se encuentre un 1 le sumara a la variable contador un uno entonces después de evaluar la condición se le suma 1 a *i* hasta que esta sea igual que la longitud de la cadena, luego se imprime en pantalla la variable contador con el acumulado de 1 que encontró en la cadena.

Al ejecutar en repl.it, se obtiene el resultado mostrado seguidamente:





The screenshot shows a web-based code editor interface. On the left, a file explorer shows 'index.html', 'script.js', and 'style.css'. The main editor displays 'script.js' with the following code:

```
6  var cadena = prompt("Introduzca un número binario");
7
8  var contador = 0; // Aquí se cuenta el número de unos
9
10 var i = 0;
11 while ( i < cadena.length ) {
12
13     if (cadena.charAt(i) === "1") {
14         contador = contador + 1;
15     }
16
17     i = i + 1;
18 }
19
20 document.write("Cadena = " + cadena + "<br><br>");
21 document.write("Contador (número de unos) = " + contador + "<br><br>");
22 document.write("FINALIZADO!");
```

On the right, the browser preview shows the output of the code:

```
Cadena = 1110110101
Contador (número de unos) = 7
FINALIZADO!
```

## 7 ÍNDICE DEL NÚMERO BINARIO

---

Este programa se encarga de mostrar el índice correspondiente a la posición de cada dígito en la cadena del número binario dado:

```
script.js
1  var cadena = prompt("Introduzca un número binario");
2
3  document.write(cadena + "<br></br>");
4
5  var indice = 0;
6  var i = 0;
7  while(i < cadena.length){
8      document.write( indice + " ");
9      indice = indice + 1;
10     i = i + 1;
11 }
12
13 document.write("<br></br>FINALIZADO");
```

https://Taller-3.voidx66.repl.co

101010101

0 1 2 3 4 5 6 7 8

FINALIZADO

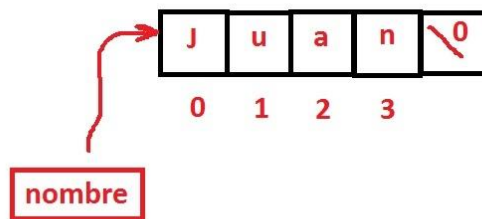
Este maneja dos variables acumuladoras las cuales se van a ir incrementando por dos razones, mientras *i* sea menor que la longitud de la cadena se imprimirá índice y un espacio, luego se incrementa índice y *i* en uno, repitiendo el proceso hasta que se deje de cumplir el `while` para finalizar el programa.



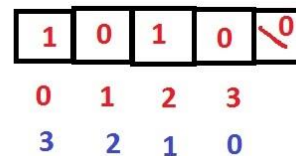
## 8 ÍNDICE DEL NÚMERO BINARIO INVERSO

A diferencia del punto anterior este mostrará un índice inverso el cual nos puede ser de mucha utilidad en el futuro cuando hablamos de conversión de bases.

```
var nombre = "Juan";
```



```
var cadena = "1010";
```



```
script.js
1  var cadena = prompt("Introduzca un número binario");
2
3  document.write(cadena + "<br><br>");
4
5  var indice = cadena.length - 1;
6  var i = 0;
7  while( i < cadena.length){
8      document.write( indice + " ");
9      indice = indice - 1;
10     i = i + 1;
11 }
12
13 document.write("<br><br>FINALIZADO");
```

https://Taller-4.voidx66.repl.co

10101010

7 6 5 4 3 2 1 0

FINALIZADO

Se introduce un dato en la variable cadena por medio de un cuadro de ventana emergente, luego se obtiene la variable indice como la longitud de la cadena menos uno y se define la variable acumuladora i como cero, mientras que i sea menor que la longitud de la cadena se imprime el indice y luego indice pasa a restar uno en su valor y i incrementa en uno, cuando indice no cumpla la condición de ser menor que la cadena es porque esta ha sido superada y por lógica indice ya debería estar en cero y así se finaliza el programa.

## 9 CONVERSIÓN DE BINARIO A DECIMAL

Utilizando lo visto en el programa anterior podremos aplicar el sistema de conversión de bases:

**" 1 0 1 1 0 "**
índice inverso

$$1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 22$$

parseInt(cadena.charAt(0))

```

script.js
1  var cadena = prompt("Introduzca un número binario");
2
3  document.write(cadena + "<br><br>");
4
5  var suma = 0;
6  var indice = cadena.length - 1;
7  var i = 0;
8
9  while( i < cadena.length){
10     suma = suma + parseInt(cadena.charAt(i))*Math.pow(2,indice);
11     indice = indice - 1;
12     i = i + 1;
13 }
14
15 document.write("<br><br>Número en base 10 = " + suma);

```

https://Cadena5-a-entero.voidx66.repl.c

10

Número en base 10 = 2

La única diferencia es que ahora se agrega la variable suma la cual tiene un valor inicial de cero y está al estar dentro del while se sumara con el resultado de la operación de la conversión de la cadena en la posición i en número multiplicado por la potencia de 2 elevado al indice.

## 10 RECURSIÓN

La recursividad es un gran recurso cuando hablamos de informática ya que este nos facilita muchas actividades, existen diferentes tipos de recursividad en la programación, pero las principales son la forma directa y la indirecta.

“La principal ventaja es la simplicidad de comprensión y su gran potencia, favoreciendo la resolución de problemas de manera natural, sencilla y elegante; y facilidad para comprobar y convencerse de que la solución del problema es correcta.

El principal inconveniente es la ineficiencia tanto en tiempo como en memoria, dado que para permitir su uso es necesario transformar el programa recursivo en otro iterativo, que utiliza bucles y pilas para almacenar las variables”<sup>1</sup>.

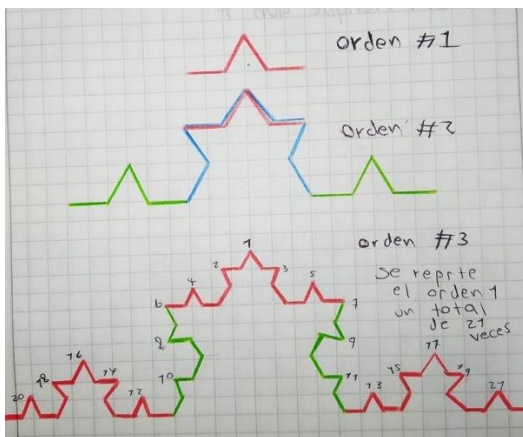


Ilustración 1 Fractales

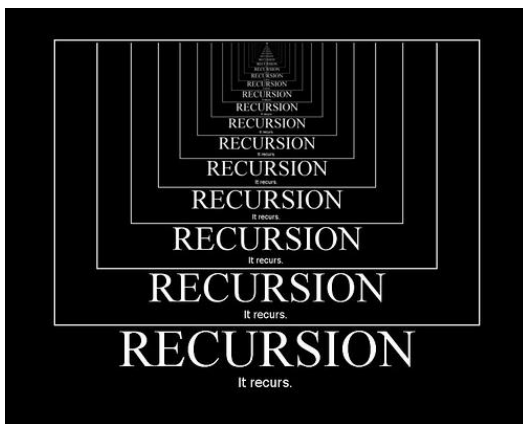
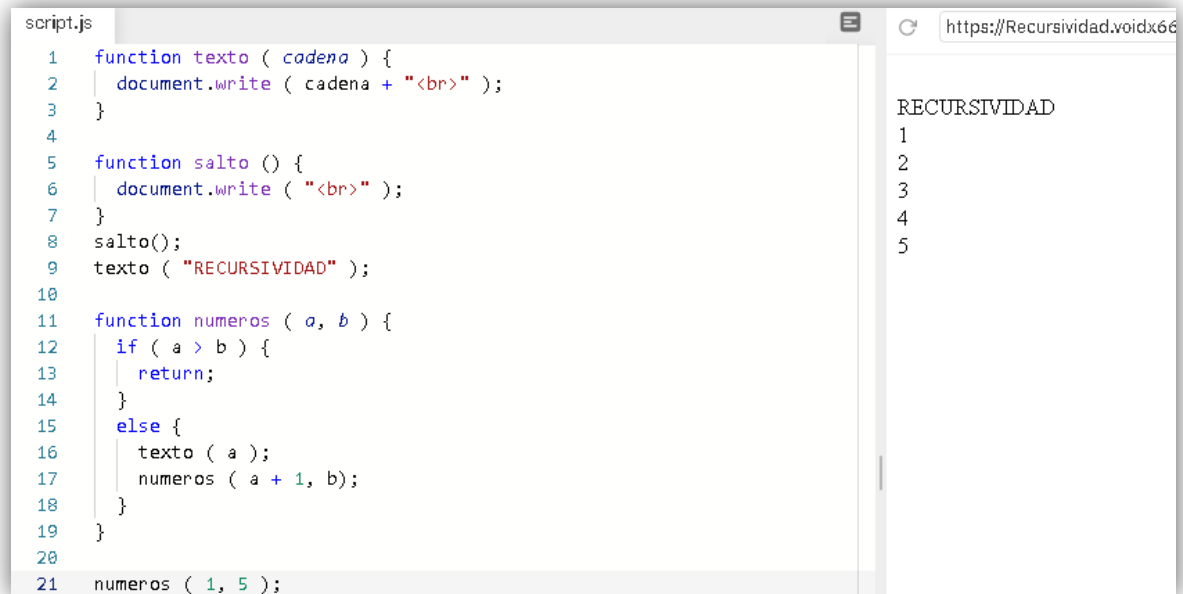


Ilustración 2 <http://mattlockyer.com/multimediamprogramming/img/recursion/recursion1.jpg>

Como nuestro entorno de trabajo es el JavaScript.

En este ejemplo vamos a recibir dos variables, una variable *a* y una *b*, si *a* es mayor que *b* este retorna, pero si esta condición no se cumple se imprime *a* y se le suma a la función *a+1* y *b* no se modifica.

Ejecución del código:



The screenshot shows a web browser window with a URL bar displaying `https://Recursividad.voidx66`. The browser's developer console or a script runner shows the execution of a JavaScript file named `script.js`. The code in the file is as follows:

```
1 function texto ( cadena ) {  
2   document.write ( cadena + "<br>" );  
3 }  
4  
5 function salto () {  
6   document.write ( "<br>" );  
7 }  
8 salto();  
9 texto ( "RECURSIVIDAD" );  
10  
11 function numeros ( a, b ) {  
12   if ( a > b ) {  
13     return;  
14   }  
15   else {  
16     texto ( a );  
17     numeros ( a + 1, b );  
18   }  
19 }  
20  
21 numeros ( 1, 5 );
```

The output of the script, visible in the browser's content area, is:

RECURSIVIDAD  
1  
2  
3  
4  
5

## 11 MULTIPLICACIÓN CON RECURSIÓN

Teniendo en cuenta el tema de la recursión vamos a crear una función que multiplique de manera recursiva.

Tenemos una función multiplicar en el cual se recibe una variable a y una variable b donde si a es igual a cero retorne cero, si no b se suma en la función y a se resta en 1.

Definimos dos variables x,y. Como ejemplo tendremos x=35,y=7

```

8  function multiplicar ( a, b ) {
9      if ( a == 0 ) {
10         return 0;
11     }
12     else {
13         return b + multiplicar ( a - 1, b );
14     }
15 }

```

```

7 + multiplicar (35 - 1, 7);
7 + multiplicar (34 - 1, 7);
7 + multiplicar (33 - 1, 7);
.
.
.
7 + multiplicar (1 - 1, 7);

```

```

7 + 0
7 + 7
7 + 14
7 + 21
.
.

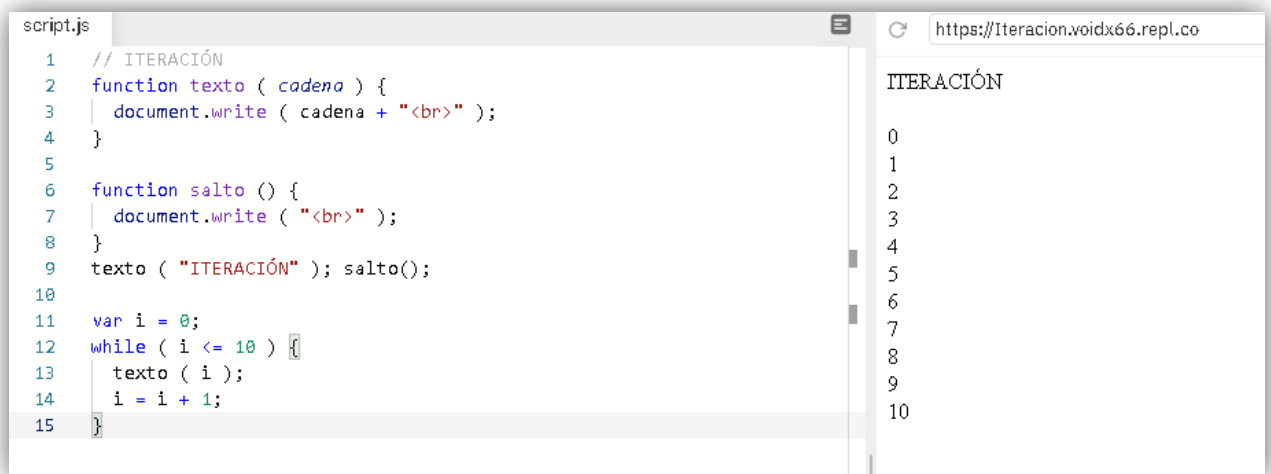
```

## 12 ITERACIÓN

---

A diferencia de la recursividad que consiste en llamarse a sí misma, la iteración como tal se basa en repetir ciertas sentencias hasta que se cumpla cierta condición<sup>2</sup>.

Un ejemplo de iteración sería el siguiente código:



```
script.js
1 // ITERACIÓN
2 function texto ( cadena ) {
3   document.write ( cadena + "<br>" );
4 }
5
6 function salto () {
7   document.write ( "<br>" );
8 }
9 texto ( "ITERACIÓN" ); salto();
10
11 var i = 0;
12 while ( i <= 10 ) {
13   texto ( i );
14   i = i + 1;
15 }
```

https://Iteracion.voidx66.repl.co

ITERACIÓN

0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10

El while hace referencia a “mientras”.

En el código mientras *i* con valor inicial de cero sea menor o igual a 10 se imprimirá en pantalla para cada *i* y ahora *i* tendrá un valor de *i* + 1. Y esto se repetirá hasta donde se cumpla la condición



## 13 CONCLUSIONES

---

El desarrollo de las temáticas elaboradas en clase utilizando el lenguaje JavaScript es una herramienta clave para comprender la programación y el desarrollo de sistemas numéricos los cuales pueden ser entendidos de manera más completa y sencilla por medio de programas.



## 14 REFERENCIAS

---

<https://repl.it>

1 <https://www.monografias.com/trabajos10/esda/esda.shtml#recu>

2 [https://codigofacilito.com/articulos/articulo\\_16\\_10\\_2019\\_16\\_22\\_35](https://codigofacilito.com/articulos/articulo_16_10_2019_16_22_35)