SISTEMAS NUMÉRICOS: Introducción a la Informática

UTP | Pereira

Juan guillermo duque montoya

OCTUBRE DE 2020

2020

# CONTENIDO

[1 CONTENIDO 1](#_Toc54244483)

[2 PRESENTACIÓN 2](#_Toc54244484)

[3 CONVERSIÓN BASADA EN DIVISIONES SUCESIVAS 9](#_Toc54244485)

[4 CONVERSIÓN EXTENDIDA 12](#_Toc54244486)

[5 POR CADA TEMA, UNA SECCIÓN SIMILAR A LAS ANTERIORES, TOMADAS DEL CLASSROOM 13](#_Toc54244487)

[6 CONCLUSIONES 14](#_Toc54244488)

[7 BIBLIOGRAFÍA 15](#_Toc54244489)

# 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.

Los números binarios pertenecen al conjunto de sistemas numéricos, El sistema de numeración Binario o código binario es utilizado para representar textos, datos o simplemente para procesar instrucciones en una computadora o en un dispositivo informático de cualquier tipo.

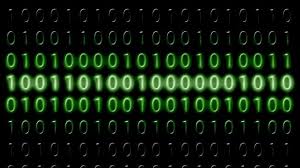


Figura 1. Números binarios

**AUTOR: Juan Guillermo Duque Montoya**

**CÓDIGO: 1004519878**

**CORREO: g.duque@utp.edu.co**

**GITHUB : https://github.com/juanguillermoduque**

# LOS NUMEROS BINARIOS

Se ha creado el siguiente código para dar una mejor explicación de lo que son los números binarios.

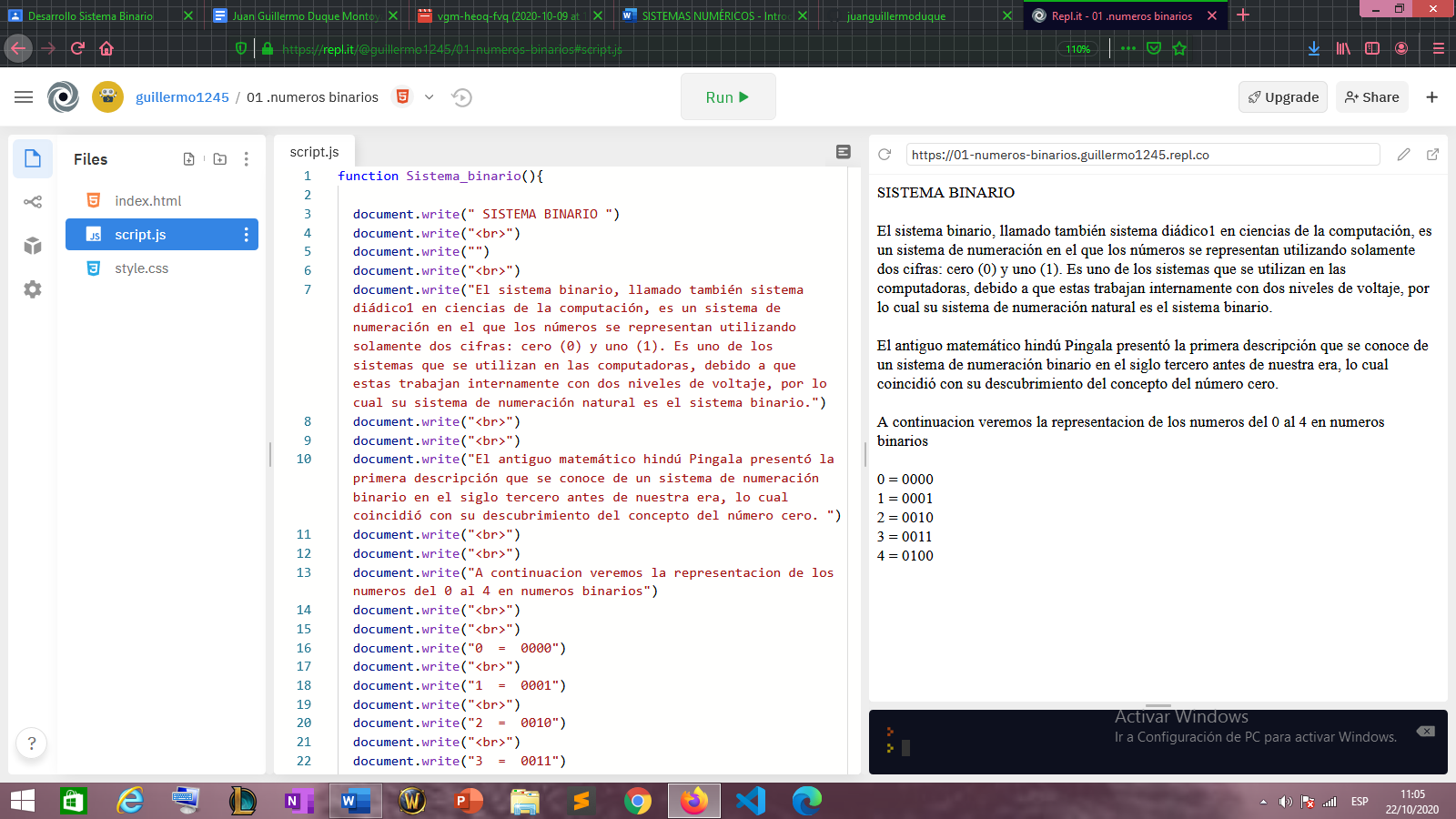


Figura2. Código números binarios 01

Como podemos ver en el anterior código, se crea una función llamada “sistema binario” adentro de esta función podemos en las palabras document.write, estas palabras significan que todo lo que este en sus paréntesis se va a escribir en el documento.

En el siguiente código veremos algunas representaciones de los números natural, es transformados a números binarios

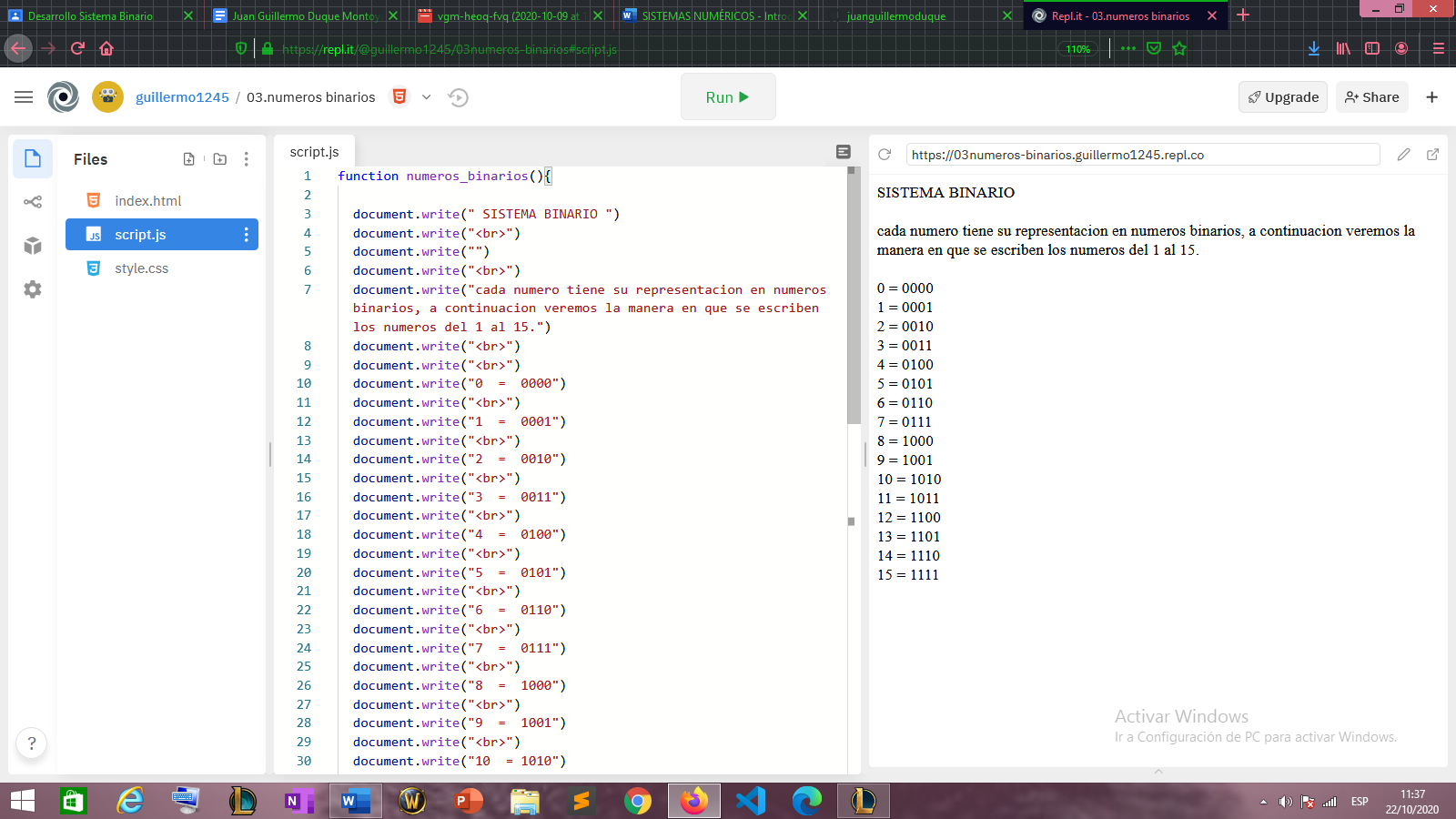


Figura 3.Codigo números binarios 02

Existen diversas formas de escribir cadenas de texto en java script por lo cual vamos a ver unos ejemplos, claro está utilizando de referencia a los números binarios.

La que manera que vamos a ver a continuación es utilizando un switch.

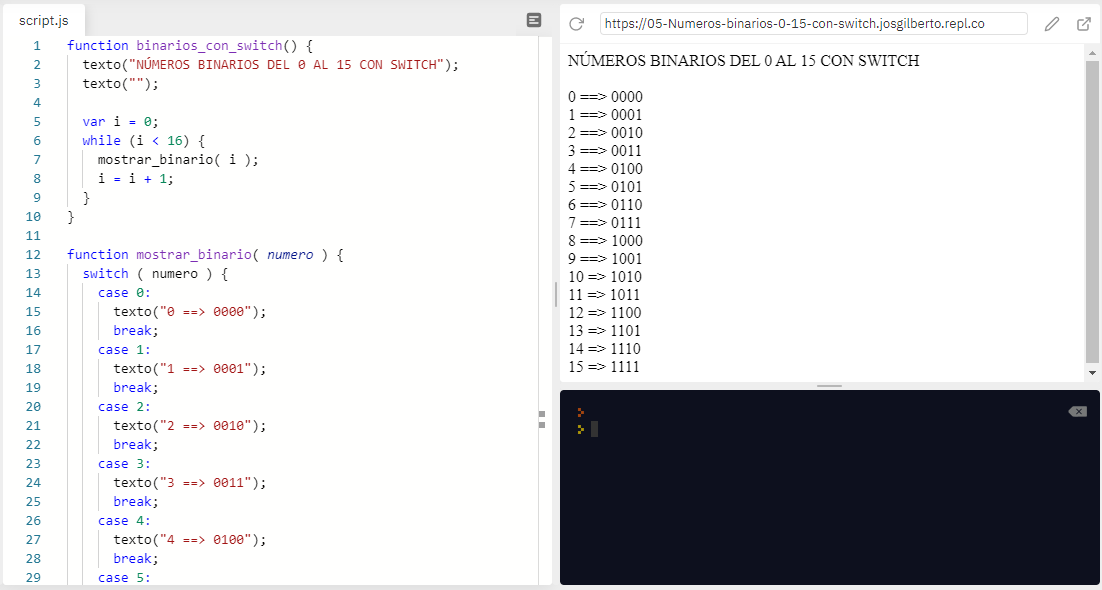


Figura 4.Codigo números binarios 03

la manera en que funciona el switch es muy sencilla de resumir, simplemente se crea un switch y dentro de el se ubican los casos que sean necesarios, cada caso tiene una condición y si la condición se cumple se ejecuta lo que esta en el interior del caso como se ve en el código anteriormente enviado.

Otra forma de representar cadenas en java script se basa en la composición de funciones, consiste en llamar una función adentro de otra función.

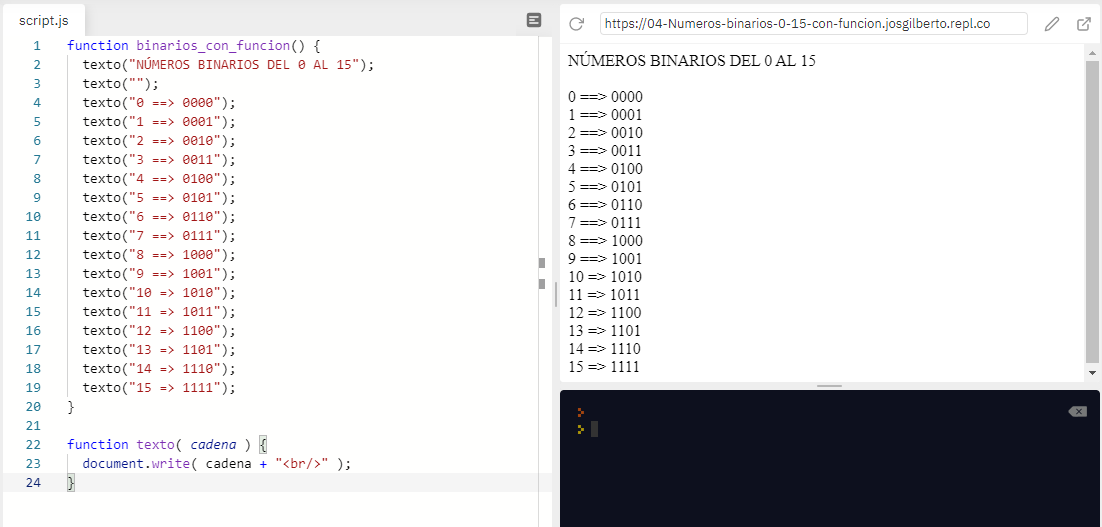


Figura 5.Codigo números binarios 04

Como vemos en la imagen se crea una función llamada texto, y luego esta función es llamada adentro de otra función que se llama binarios\_con\_funcion, esto sucede por que muchas veces nos ayuda a ahorrar líneas de código lo que significa menos trabajo y el mismo resultado, en este caso es más fácil escribir texto, a escribir document. Write

Como es de inferir los números binarios son muy importantes para la tecnología actual, ya que muchas de las instrucciones computarizadas se basan en números binarios por esta razón hemos decidido entrar más a fondo, y en las siguientes paginas se van a presentar una serie de problemas y soluciones todo alrededor de los números binarios

**TALLER SOBRE NUMEROS BINARIOS**

A continuación, veremos 4 talleres que fueron propuestos para explicar durante la clase

**TALLER 1 ---------** Algoritmo 1. Leer una cadena por teclado 2. Recorrer la cadena desde la posición 0, hasta su última posición 3. Para cada carácter tomado de la cadena, se evalúa si vale cero o si vale 1 4. Si vale 0, escribir en la pantalla Cero 5. Si vale 1, escribir en la pantalla Uno EJEMPLO: Entrada: 0110 Salida: Cero Uno Uno Cero.

**SOLUCIÓN TALLER 1**

<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <title>Taller 1</title>

</head>

<body>

    <script type="text/javascript">

        var cadena = prompt("ingrese un numero binario");

        var salida = "";

        var i = 0;

        while ( i < cadena.length) {

            if (cadena.charAt(i) === "0") {

            salida = salida + "Cero ";

        }

            if (cadena.charAt(i) === "1") {

            salida = salida + "Uno ";

        }

            i = i + 1;

        }

        alert(salida);

    </script>

</body>

</html>

**TALLER 2** --------- Algoritmo 0. Inicializar un contador en 0 (contador = 0) 1. Leer una cadena por teclado 2. Recorrer la cadena desde la posición 0, hasta su última posición 3. Para cada carácter tomado de la cadena, se evalúa si vale cero o si vale 1 4. Si vale 1, se suma 1 al contador (contador = contador + 1) 5. Una vez termine el ciclo, presentar en una ventana emergente el valor correspondiente al número total de Unos contenidos en el número binario EJEMPLO: Entrada: 0110 Salida: 2

**SOLUCIÓN TALLER 2**

<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <title>Taller 2</title>

</head>

<body>

    <script type="text/javascript">

        var cadena = prompt("ingrese un numero binario");

        var contador = 0;

        var i = 0;

        while ( i < cadena.length) {

            if (cadena.charAt(i) === "0") {

                contador = contador + 0;

        }

            if (cadena.charAt(i) === "1") {

                contador = contador + 1

        }

            i = i + 1;

        }

        alert("El total de 1 es:  " + contador);

    </script>

</body>

</html>

**TALLER 3** ---------- Algoritmo 0. Inicializar un contador en 0 (contador = 0) 1. Leer una cadena por teclado 2. Recorrer la cadena desde la posición 0, hasta su última posición 3. El contador se corresponde con el índice. Mostrar el índice en una ventana emergente 4. Incrementar el contador (contador = contador + 1) 5. Una vez termine el ciclo, el programa finaliza EJEMPLO: Entrada: 011011 Salida: 0 1 2 3 4 5

**SOLUCIÓN TALLER 3**

<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <title>Taller 3</title>

</head>

<body>

    <script type="text/javascript">

        var cadena = prompt("ingrese un numero binario");

        var contador = 0;

        var i = 0;

        while ( i < cadena.length) {

            alert (contador);

            if (cadena.charAt(i) === "0") {

                contador = contador + 1;

        }

            if (cadena.charAt(i) === "1") {

                contador = contador + 1;

        }

            i = i + 1;

        }

    </script>

</body>

</html>

**TALLER 4** ---------- Algoritmo 1. Leer una cadena por teclado 2. Inicializar contador en longitud de cadena - 1 2. Recorrer la cadena desde la posición 0, hasta su última posición 3. El contador se corresponde con el índice. Mostrar el índice en una ventana emergente 4. DECREMENTAR el contador (contador = contador - 1) 5. Una vez termine el ciclo, el programa finaliza

**SOLUCIÓN TALLER 4**

<!DOCTYPE html>

<html lang="es">

<head>

    <meta charset="UTF-8">

    <title>Taller 4</title>

</head>

<body>

    <script type="text/javascript">

        var cadena = prompt("ingrese un numero binario");

        var contador = cadena.length - 1 ;

        var i = -1;

        while ( i < cadena.length) {

            alert (contador);

            if (cadena.charAt(i) === "0") {

                contador = contador - 1;

        }

            if (cadena.charAt(i) === "1") {

                contador = contador - 1;

        }

            i = i + 1;

        }

    </script>

</body>

</html>

# COMPUERTAS BINARIAS

Las compuertas binarias son las que comparan los números binarios y dependiendo de la compuerta se da un retorno lo entenderemos mejor con el siguiente código.

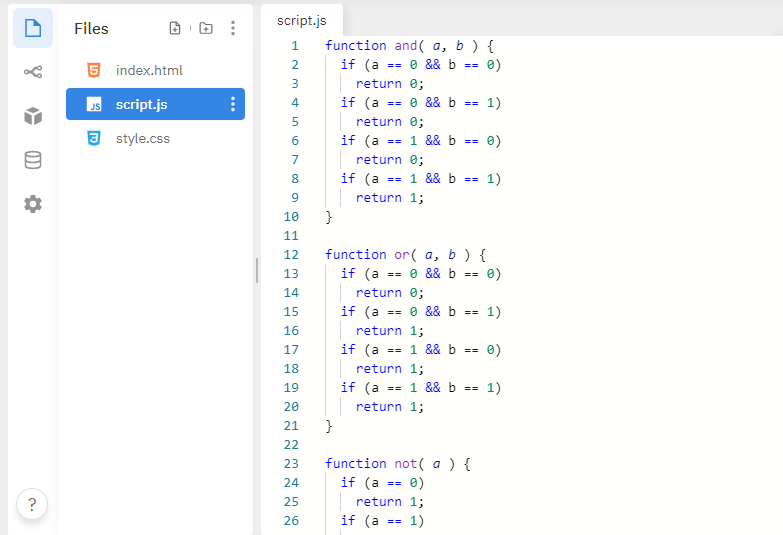


Figura 6. Compuertas binarias 01

Como vemos en el código se crean varias funciones cada función contiene las diferentes compuertas binarias que existen, como vemos existen 3 compuertas binarias la primera es el and, la segunda el or y la tercera el not, como se ve en el código cada función retorna un valor dependiendo del valor de a y b, de esta manera funcionan las compuertas binarias

Al ejecutar el anterior código se vera la siguiente interfaz

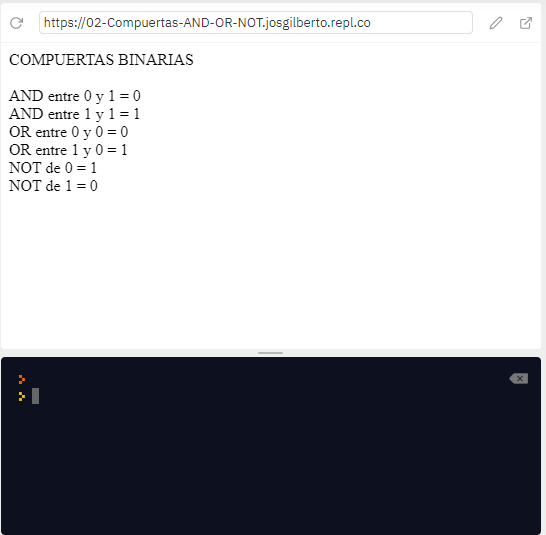


Figura 7. Compuertas binarias 02

# CONVERSIÓN BASADA EN DIVISIONES SUCESIVAS

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

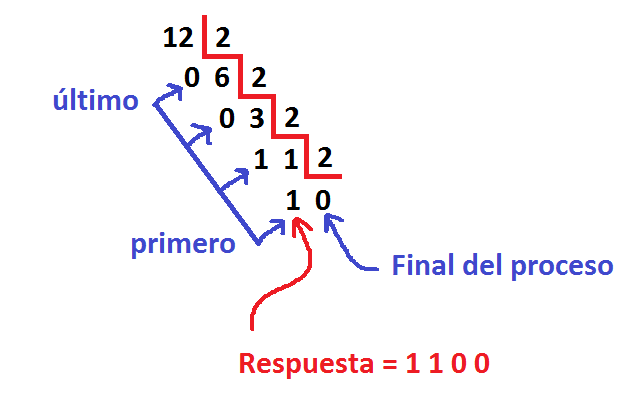


Figura 8. Conversión en división sucesiva

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 presenta 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.

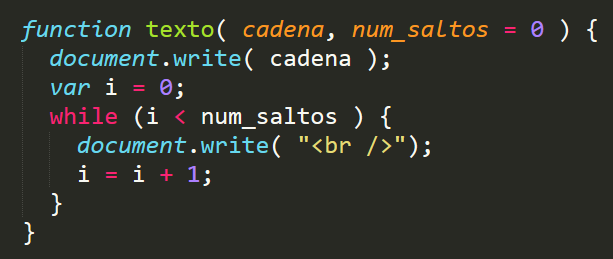


Figura 9. Conversión en división sucesiva 02

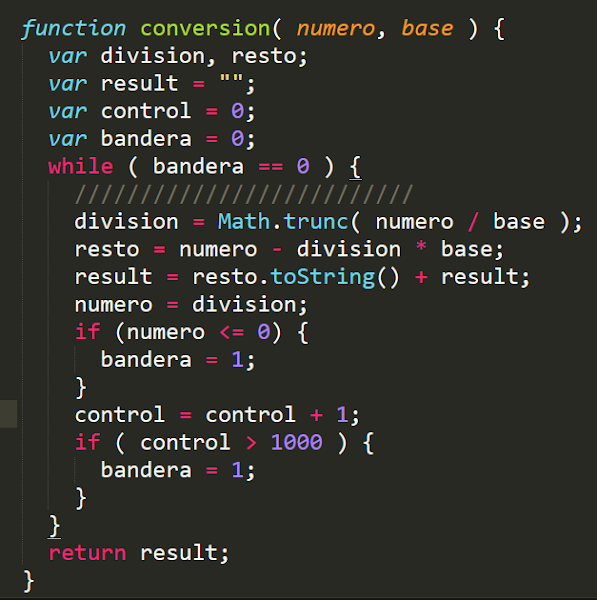


Figura 10. Conversión en división sucesiva 03

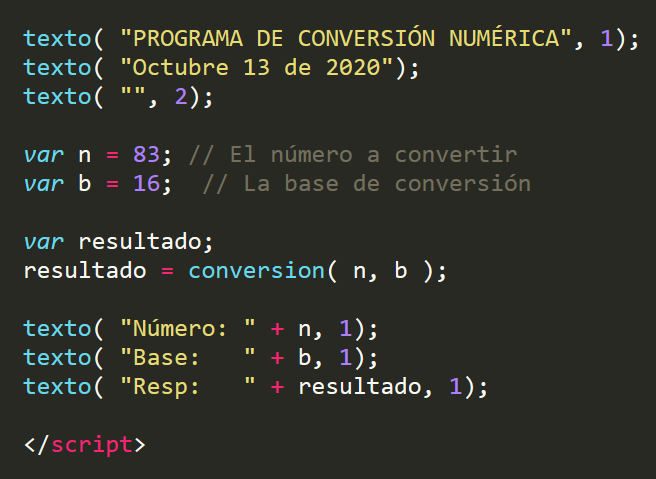


Figura 11. Conversión en división sucesiva 04

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

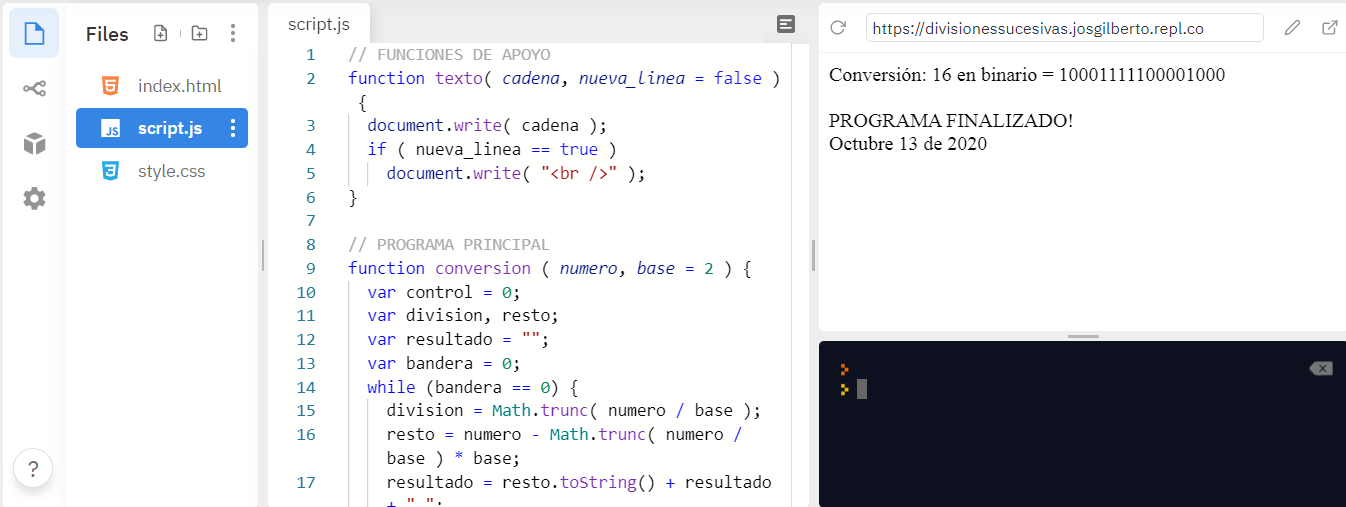


Figura 12. Conversión en división sucesiva 05

Es muy interesante recalcar el uso se funciones y la composición de ellas, y saber lo fácil y rápido que puede ser llegar elaborar un programa que realice operaciones que a simple vista parecen muy avanzadas, lo único que se necesita es lógica y un poco de conocimiento en programación.

# CONVERSIÓN EXTENDIDA

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

Es una forma de convertir un nuero entero a un número binario, pero de forma extendida, los resultados se visualizan del 10 al 15 con las letras A B C D E F.

A continuación, veremos un código donde se ponen a prueba la conversión extendida de números binarios.

Para empezar, vamos a crear nuestra famosa función llamada texto, la cual nos permitirá ahorrar tiempo y espacio de código

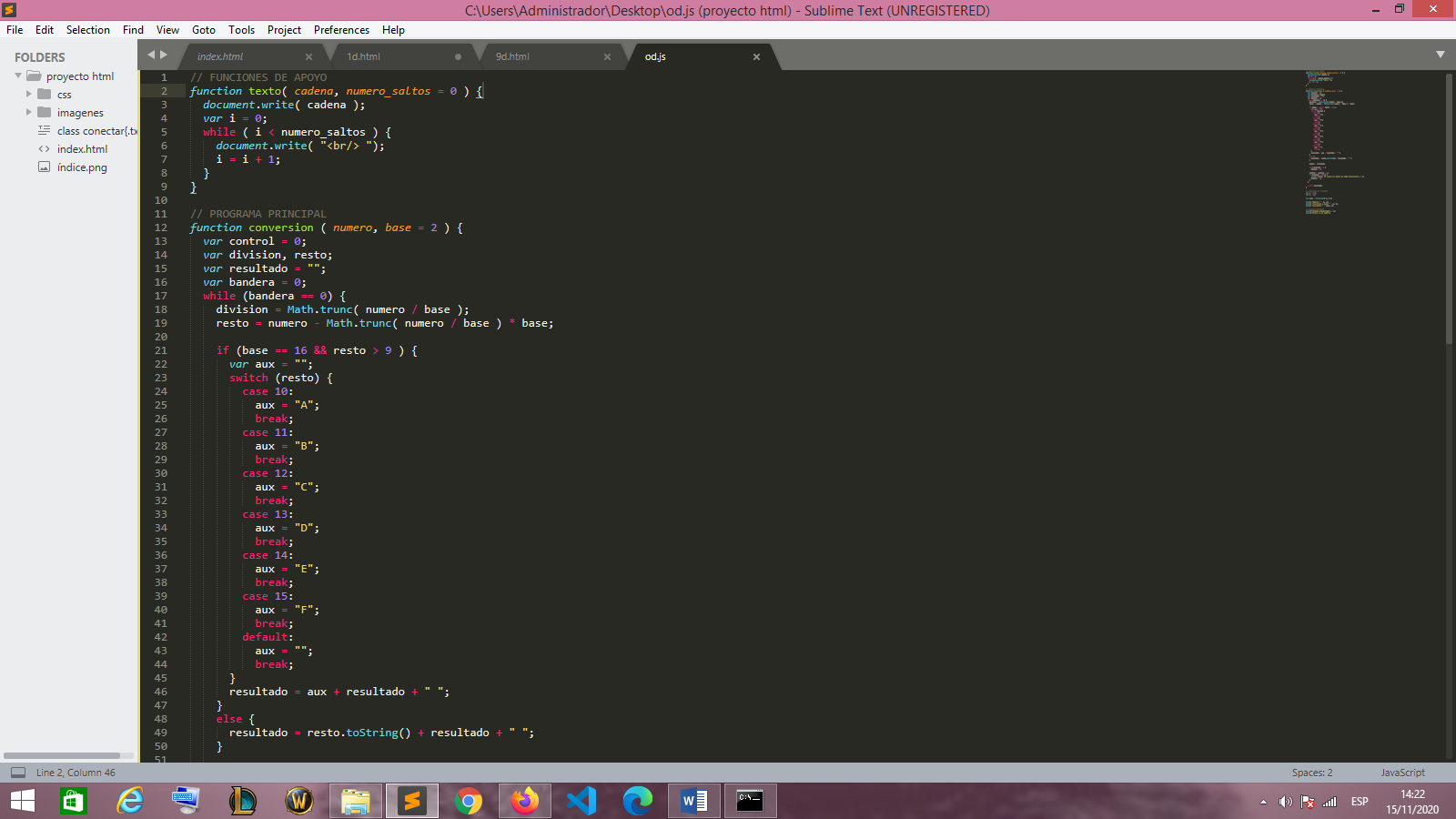


Figura 13. Conversión extendida 01

Una vez escrita nuestra función texto, procederemos a crear una nueva función a la que denominaremos conversión en esta se realizara todo lo respectivo a la conversión extendida, la función es un poco larga y para mejor comprensión la he dividido en dos partes.

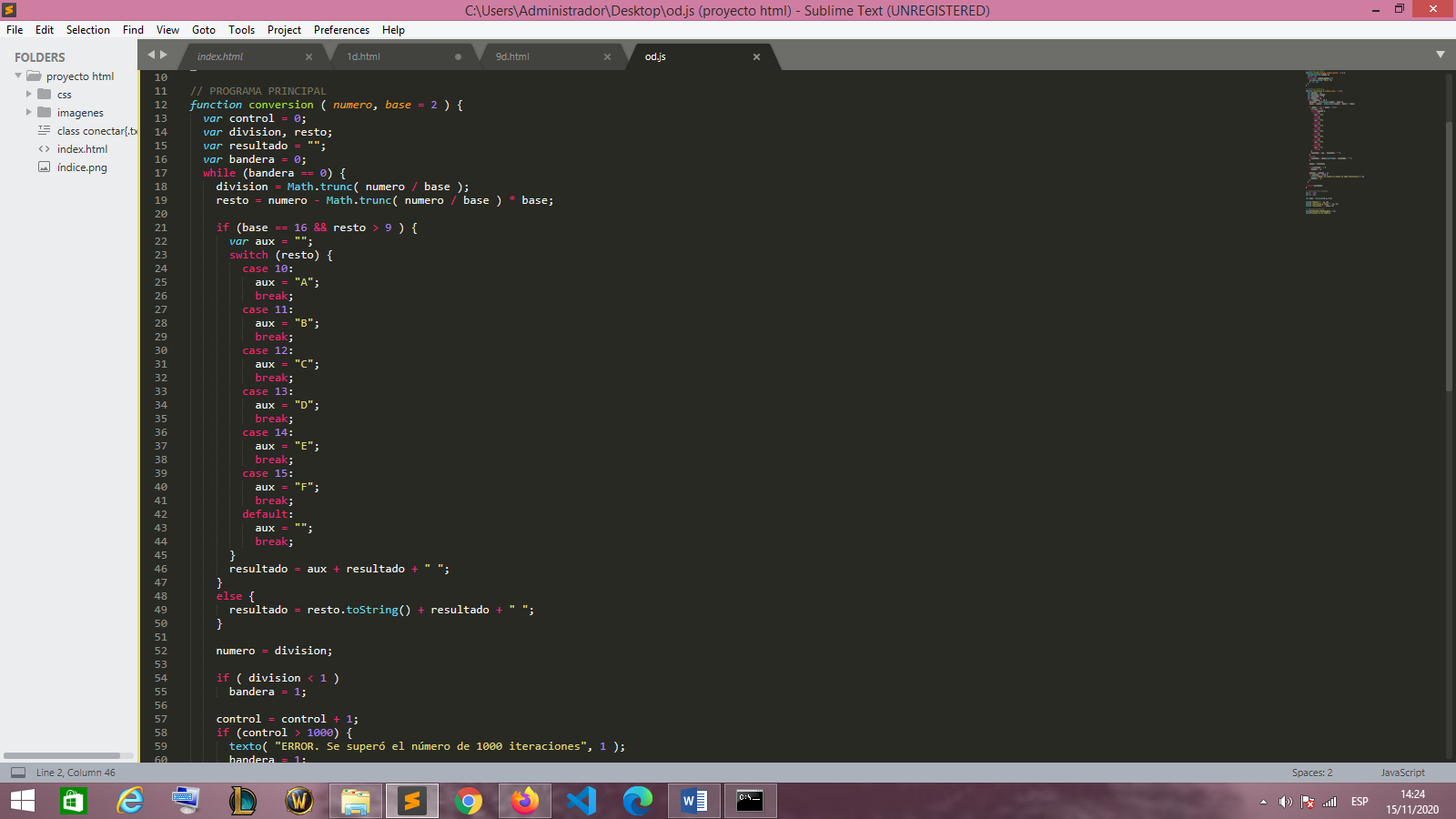


Figura 14. Conversión extendida 02

Esta es la primera parte y como vemos el número que entra es dividido por la base que en este caso también entra como parámetro, se crea una nueva variable que almacenara el residuo resultado de dicha división, se crea un condicional que devuelve los resultados del 10 al 15 en forma de letras.

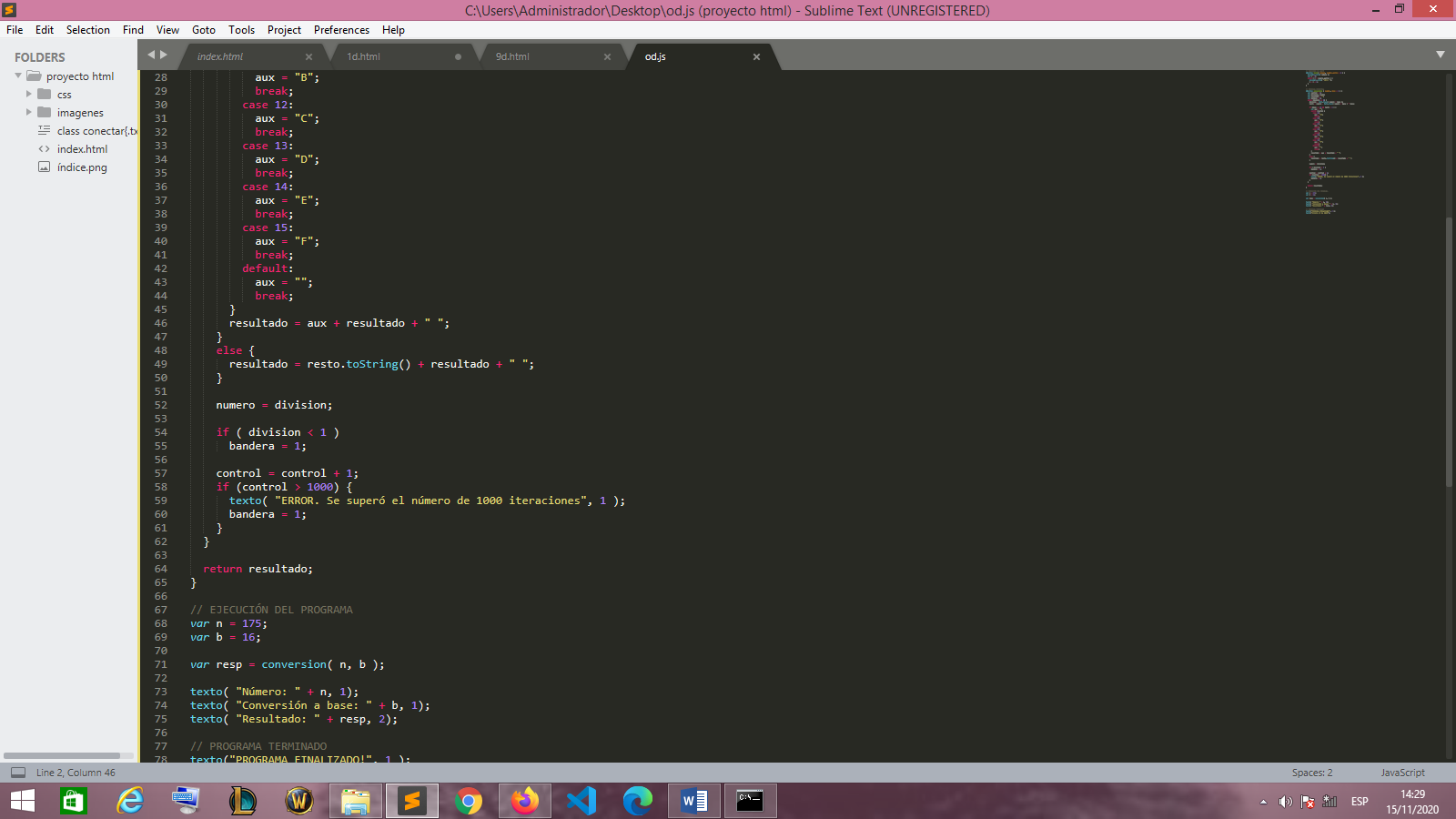


Figura 15. Conversión extendida 03

En la segunda parte de la función conversión, se suma el resultado con la variable obtenida del condicional pasado, y además se está controlando el límite de iteraciones que puede soportar la función y como lo dice la condicional, el límite de iteraciones es de 1000.

Para terminar, veremos unas ejecuciones para dar un ejemplo grafico de cómo funciona las funciones valga la redundancia, las ejecuciones que realizaremos son las siguientes.

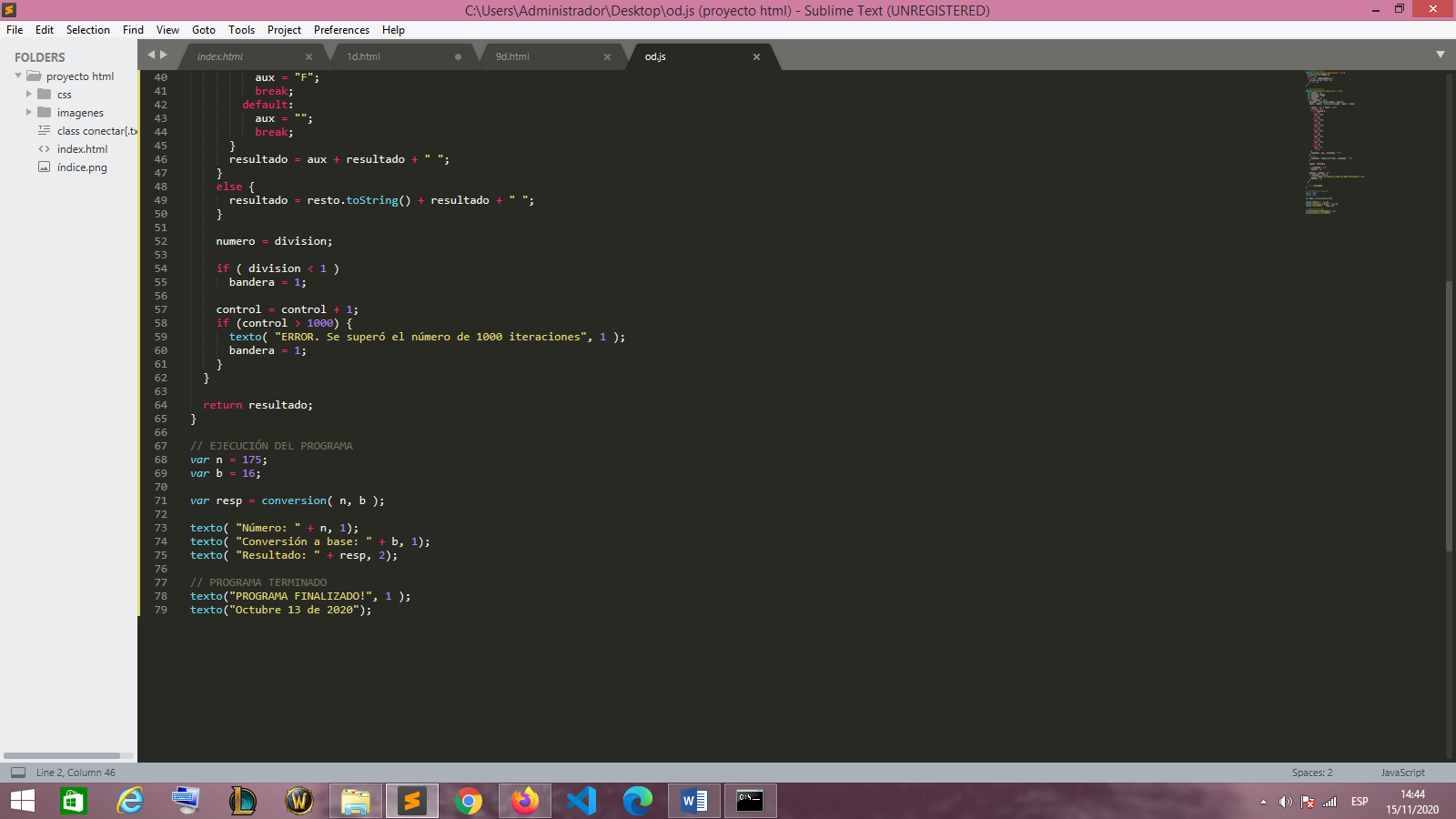


Figura 16. Conversión extendida 04

Como vemos invocamos a la función y les mandamos sus respectivas bases y exponente como argumentos.

Al ejecutarlo en repl.it el resultado es el siguiente:

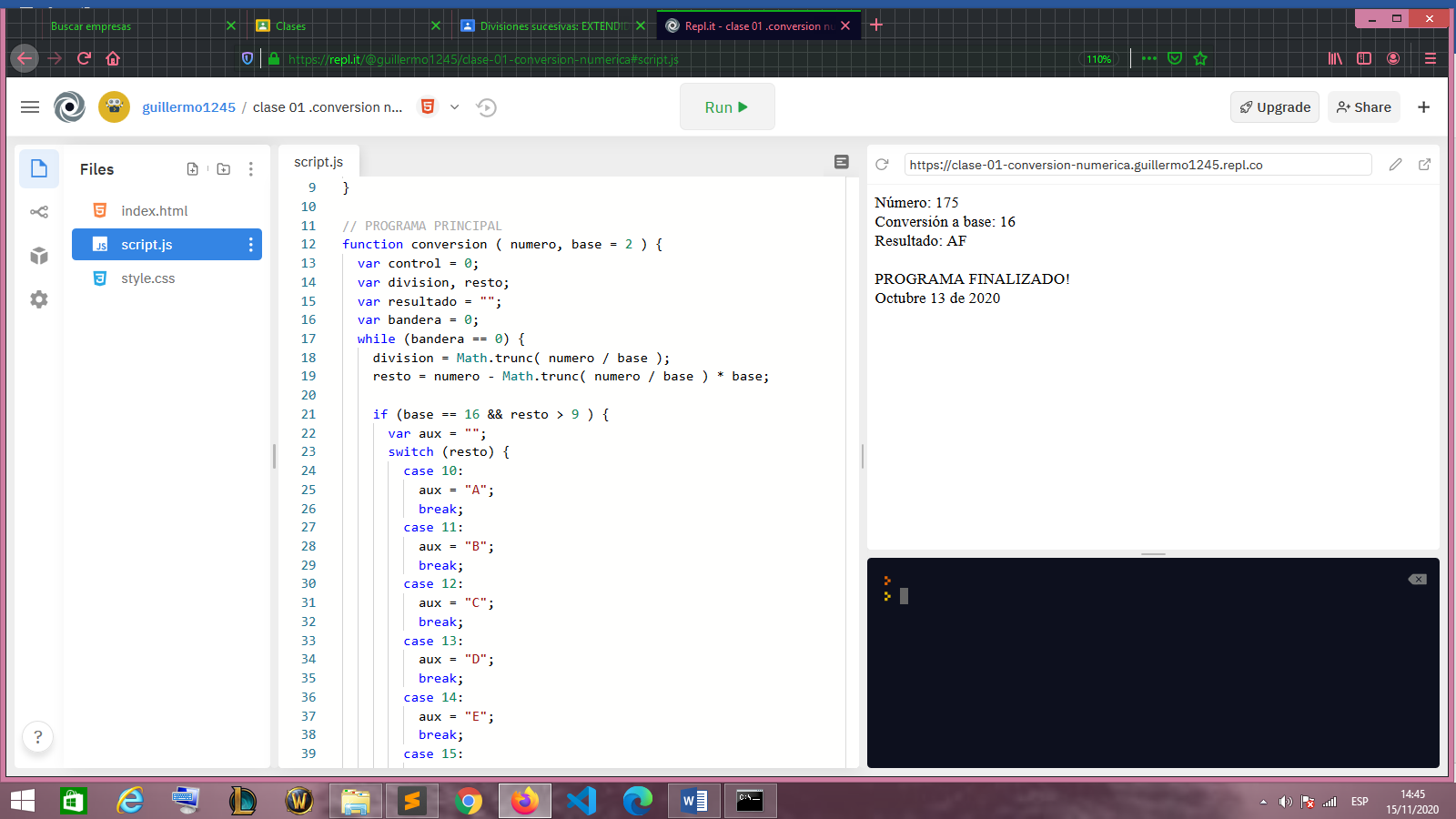


Figura 17. Conversión extendida 05

# RECURSIVIDAD

Podemos definir a **la recursividad** como un método de definir un proceso a través del uso de premisas que no dan más información que el método en sí mismo o que utilizan los mismos términos que ya aparecen en su nombre, por ejemplo, cuando se dice que la definición de algo es ese algo mismo.

Otra buena forma de definir la recursividad, pero esta vez más enfocado al ámbito computacional de la programación es un ciclo o bucle, ya que son acciones que se repiten una y otra vez hasta algún determinado tiempo, incluso existen acciones recursivas que nunca cesan.

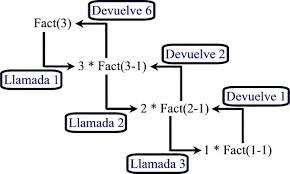


Figura 18. Recursividad

En la imagen podemos observar la recursividad en su expresión más codificada, vemos como existen 3 llamas y cada llamada tiene una devolución, pero al completarse el circuito, se vuelve a reiniciar el proceso, convirtiéndose en un ciclo hecho por el llamado recursivo de la función fac.

Para entender mejor el tema de recursividad, primero debemos repasar otros conceptos que son claves a la hora de ser recursivos en un código.

El primero es la **SECUENCIALIDAD**: Básicamente es crear por medio de una programación manual generar una secuencia, normalmente esta secuencia es numérica, como lo veremos en el siguiente código.

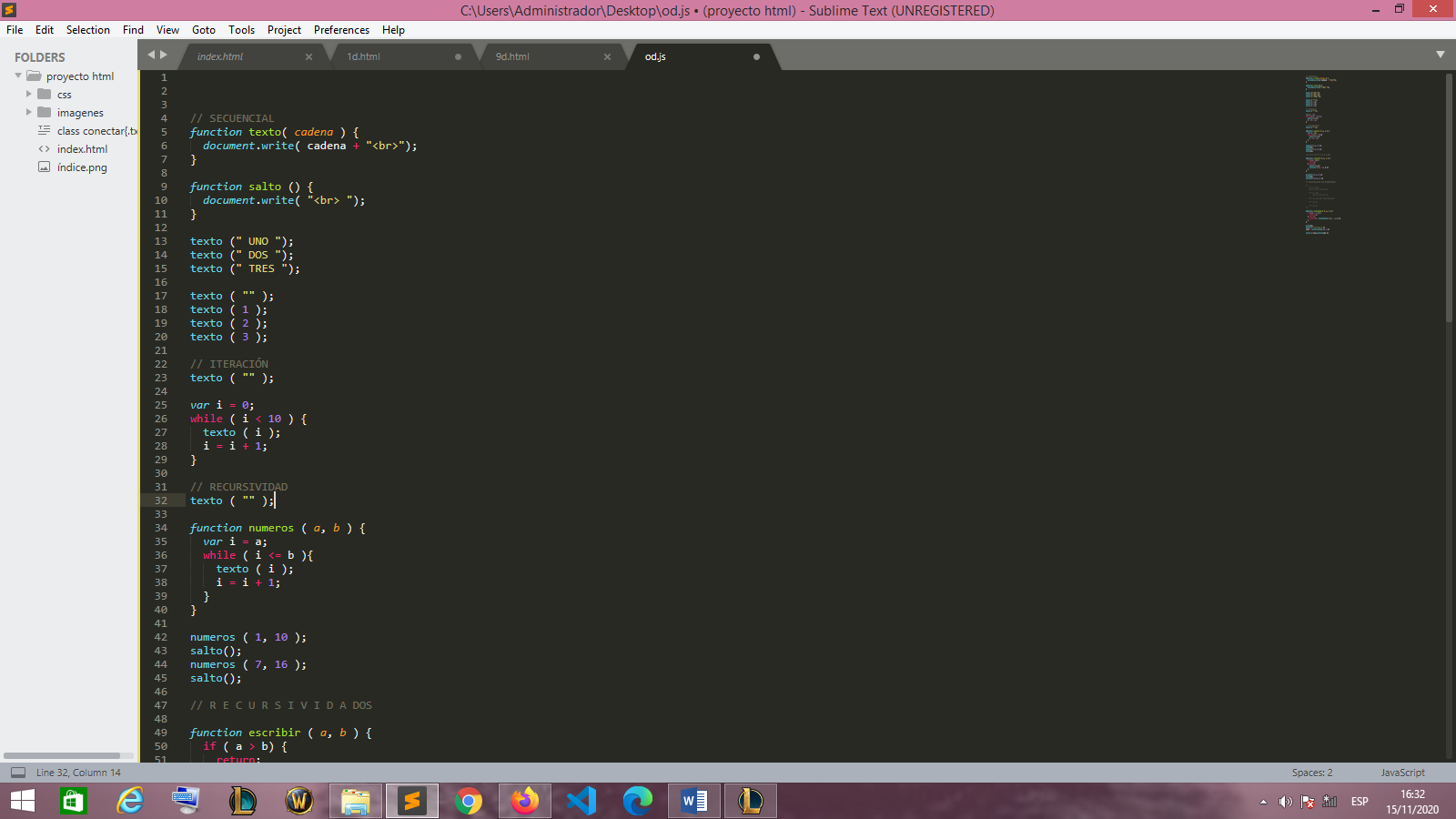


Figura 19. Recursividad 02

Creamos una función texto y la invocamos las veces que necesitemos que un número sea imprimido. Al ejecutar el código veremos la siguiente interfaz.

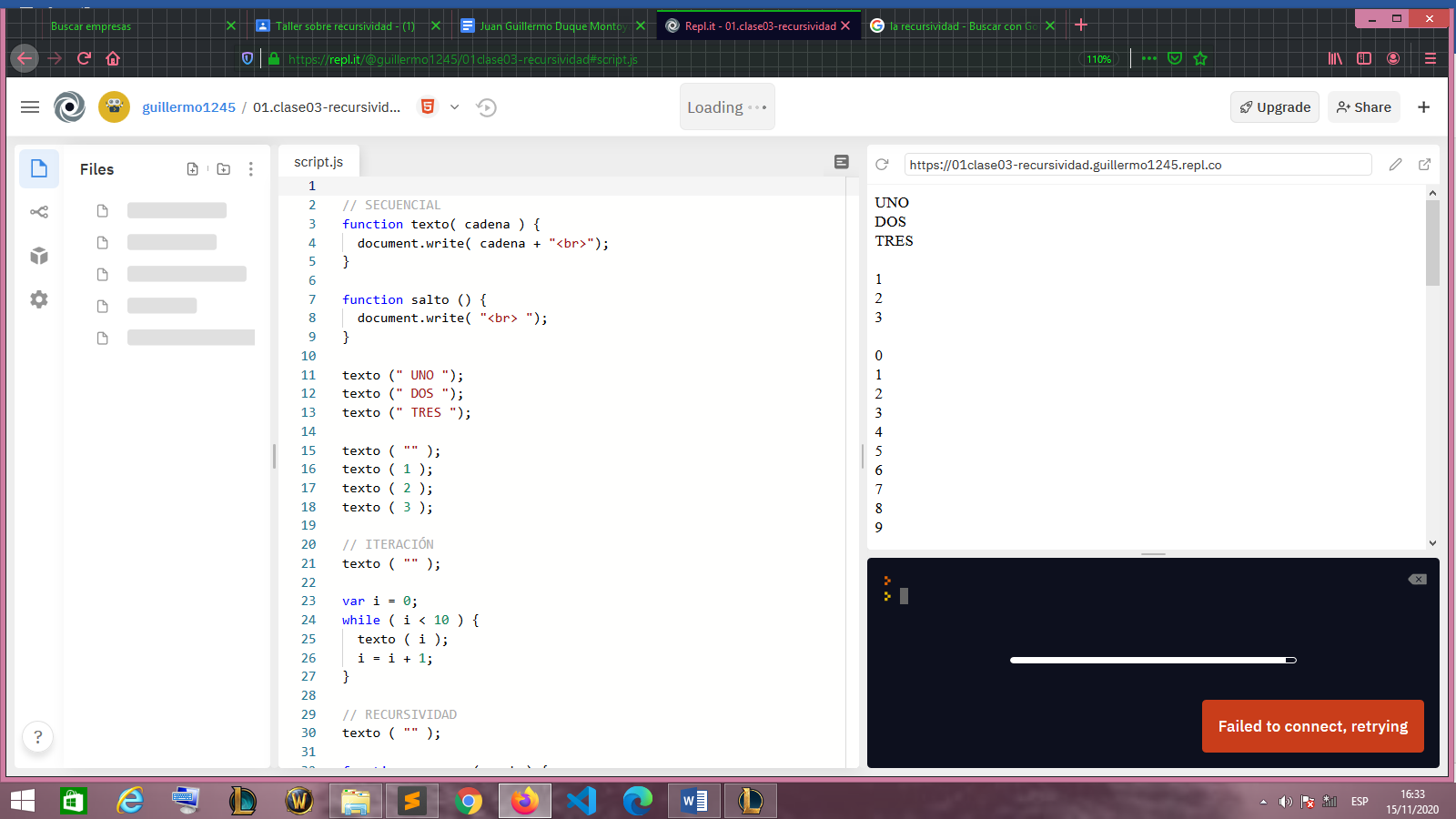


Figura 20. Recursividad 03

El segundo punto necesario para aprender mejor sobre la recursividad es, **LA ITERACION:** en pocas palabras y de manera muy entendible es utilizar ciclos para generar secuencias, se le conoce como iteración, porque se relacionan múltiples veces los elementos que hacen parte del ciclo, y cada vez que el ciclo se repite se modifica alguno de estos objetos para que la secuencia se pueda generar, lo veremos en el siguiente código.



Figura 21. Recursividad 04

Como podemos ver el código es mucho más pequeño pero su funcionalidad es más efectiva, ya que nos ahorramos mucho tiempo y espacio, la ejecución la veremos a continuación.

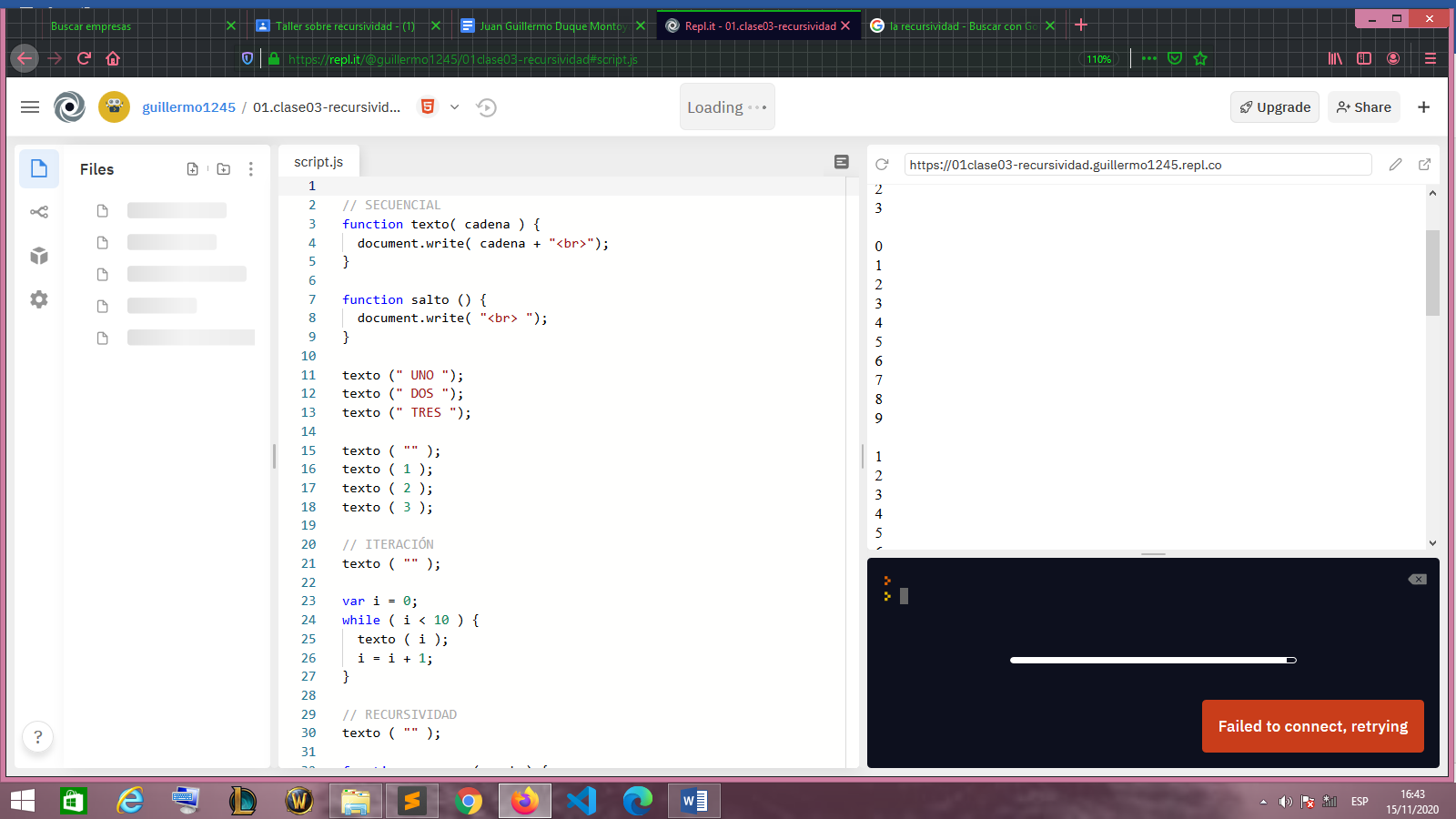


Figura 22. Recursividad 04

Ahora que tenemos una idea base sobre lo que es la recursividad, la importancia que tiene a la hora de programar, y tenemos claros los conocimientos básicos necesarios para comprender la recursividad. Veremos unos ejercicios que han sido propuestos para entender mejor el tema

EJEMPLO RECURSIBIDAD 1:

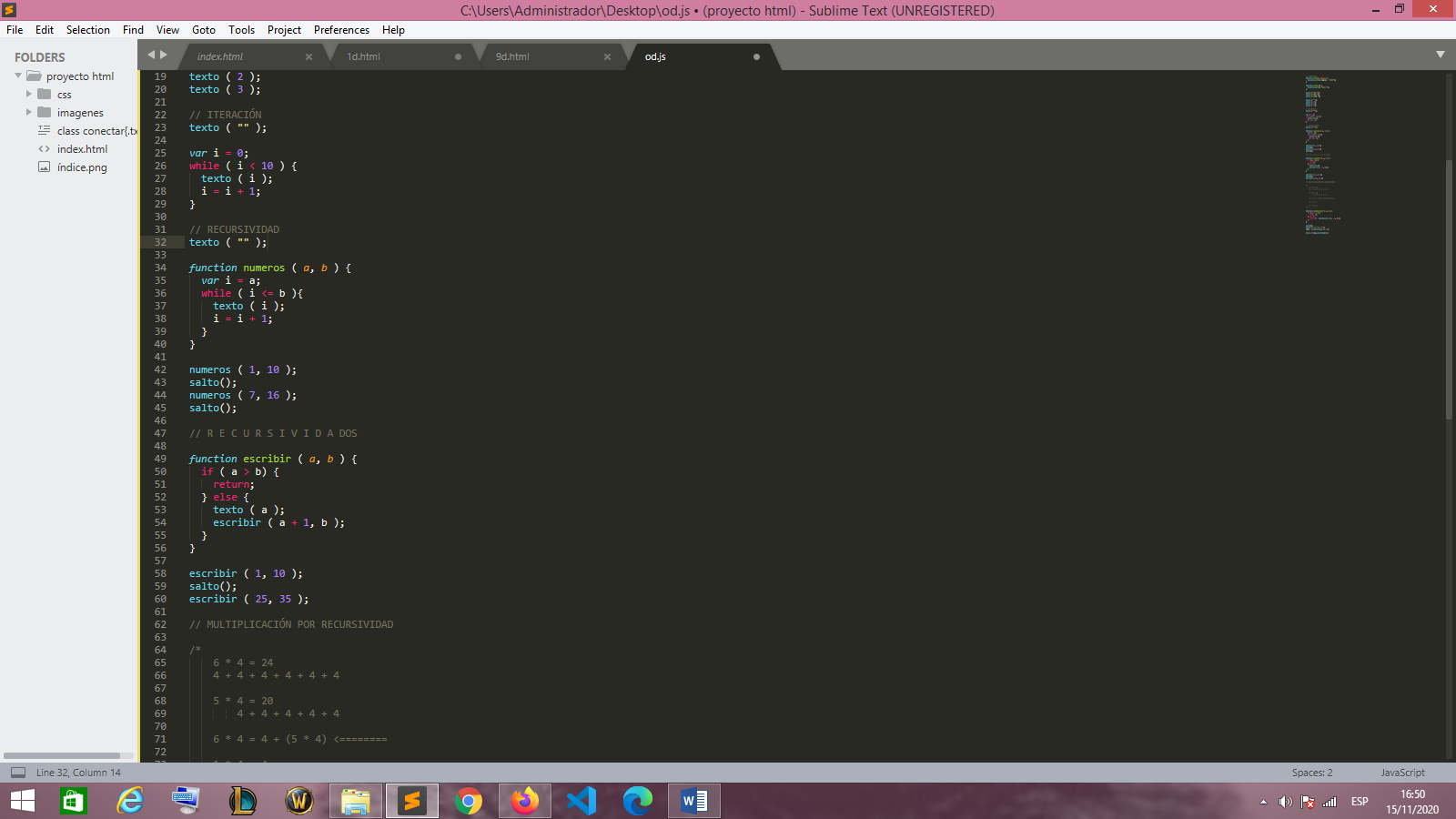


Figura 23. Recursividad 05

Utilizando la recursividad queremos crear una función que imprima los números desde un valor a hasta un valor b, como lo veremos a continuación.

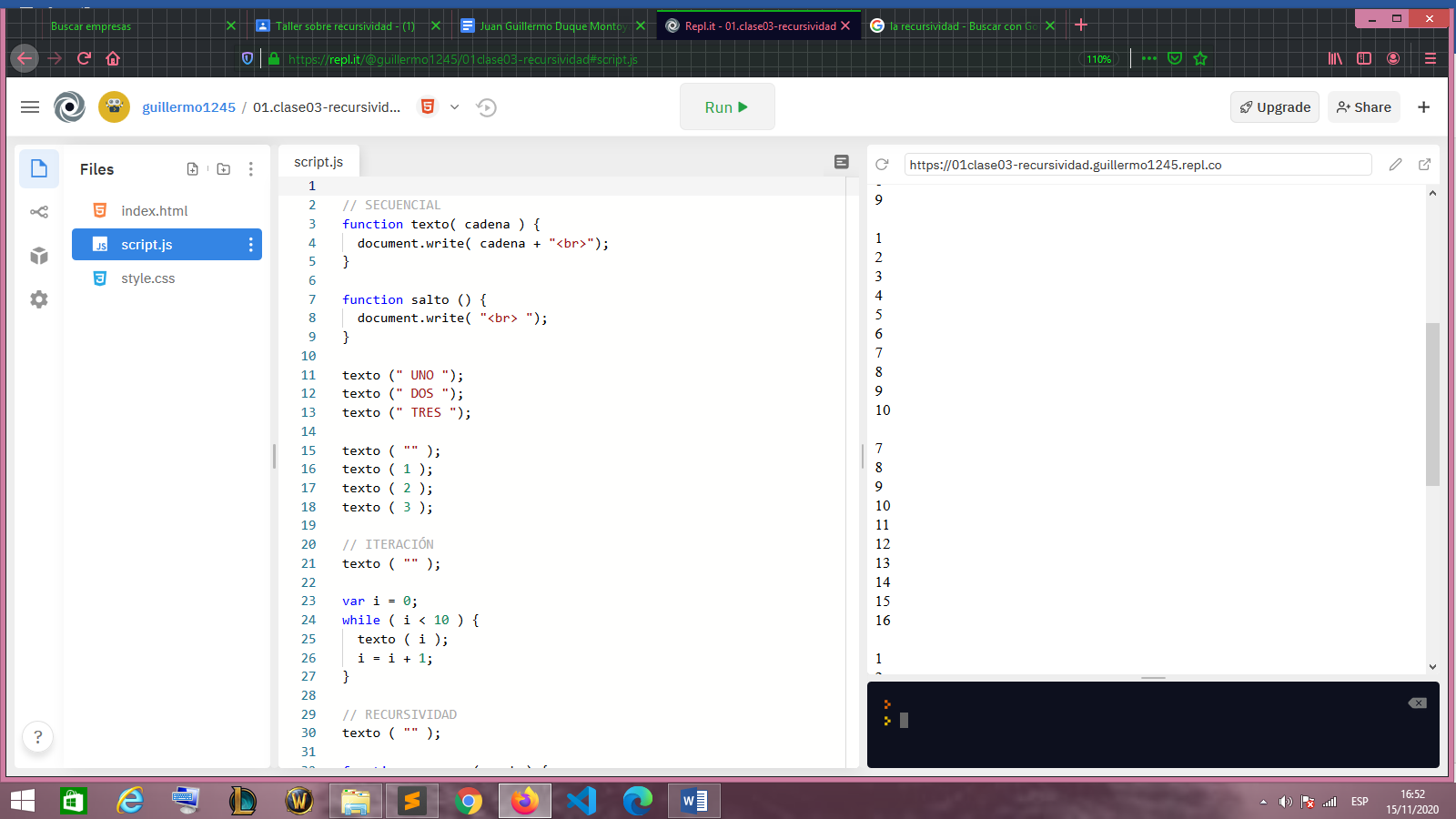


Figura 24. Recursividad 06

EJEMPLO RECURSIBIDAD 2:

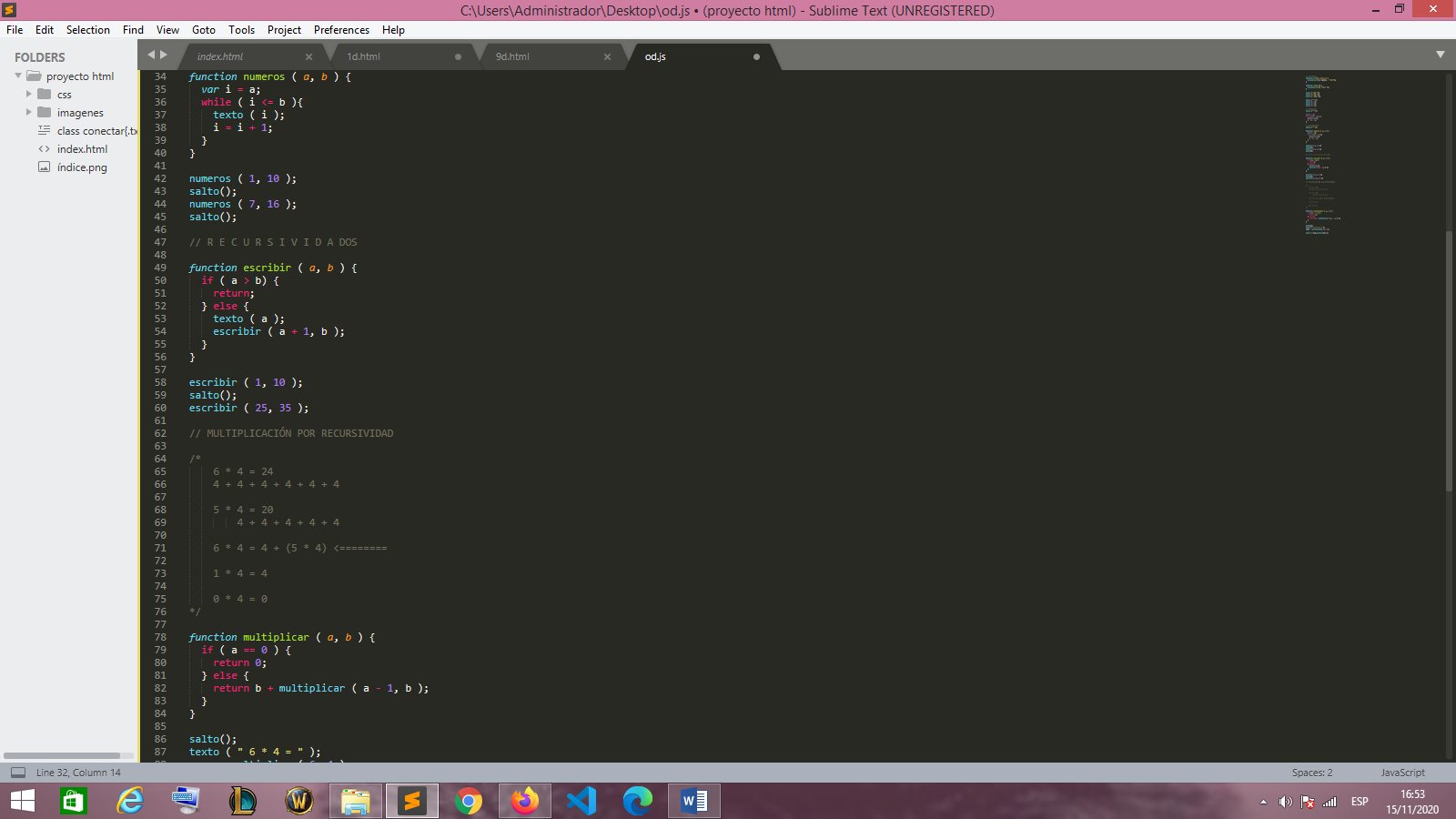


Figura 25. Recursividad 07

Una forma diferente de hacer el mismo procedimiento visto en el ejemplo 1, claro está también utilizando la recursividad, la solución al ejemplo es:

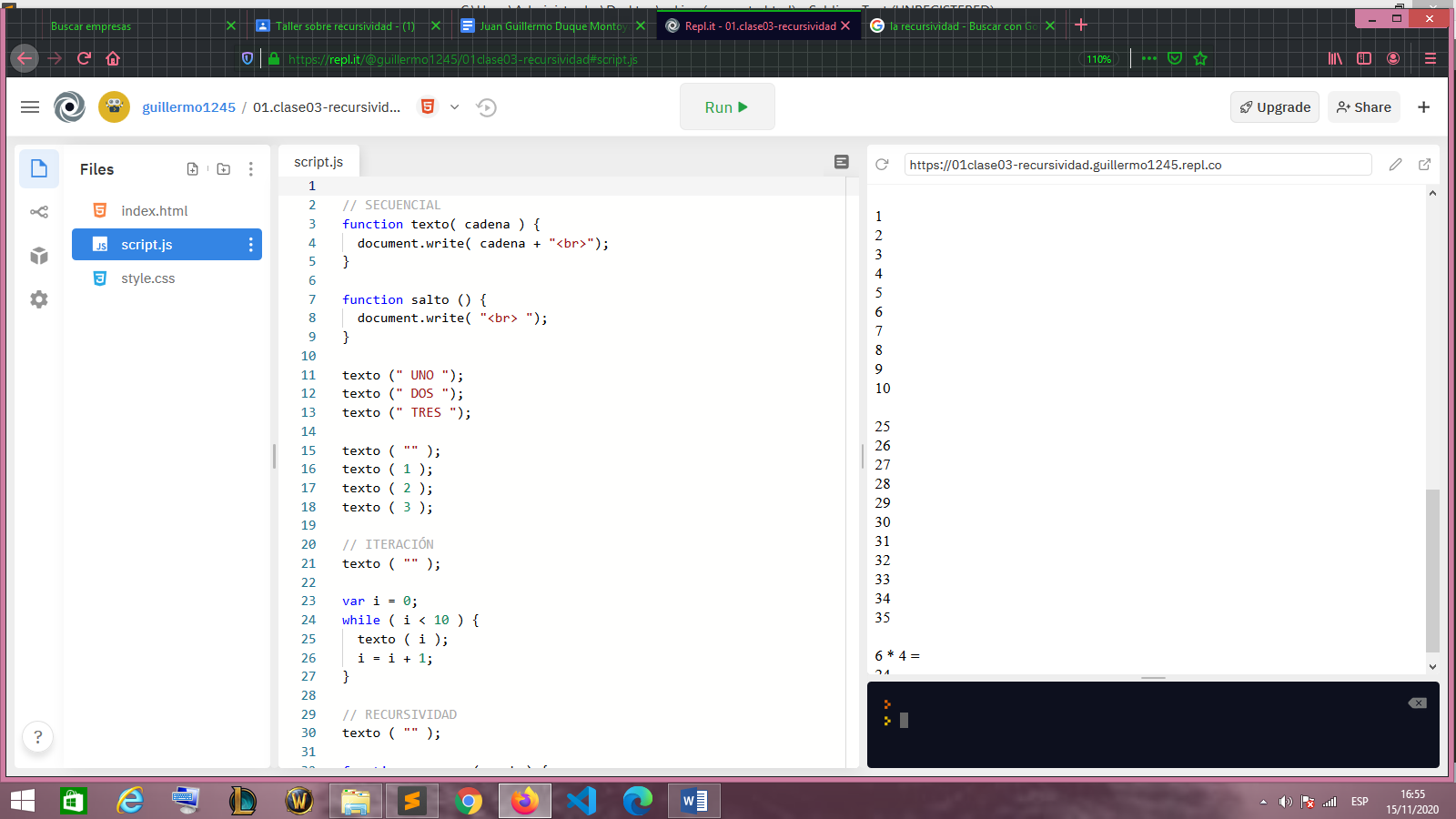


Figura 26. Recursividad 08

Ahora que tenemos claro tanto los conceptos de la recursividad, como su puesta en práctica a la hora de programar, se propusieron algunos ejercicios.

T A L L E R ( recursividad )

1. factorial ( n ) = n \* factorial ( n - 1)

**SOLUCION:**

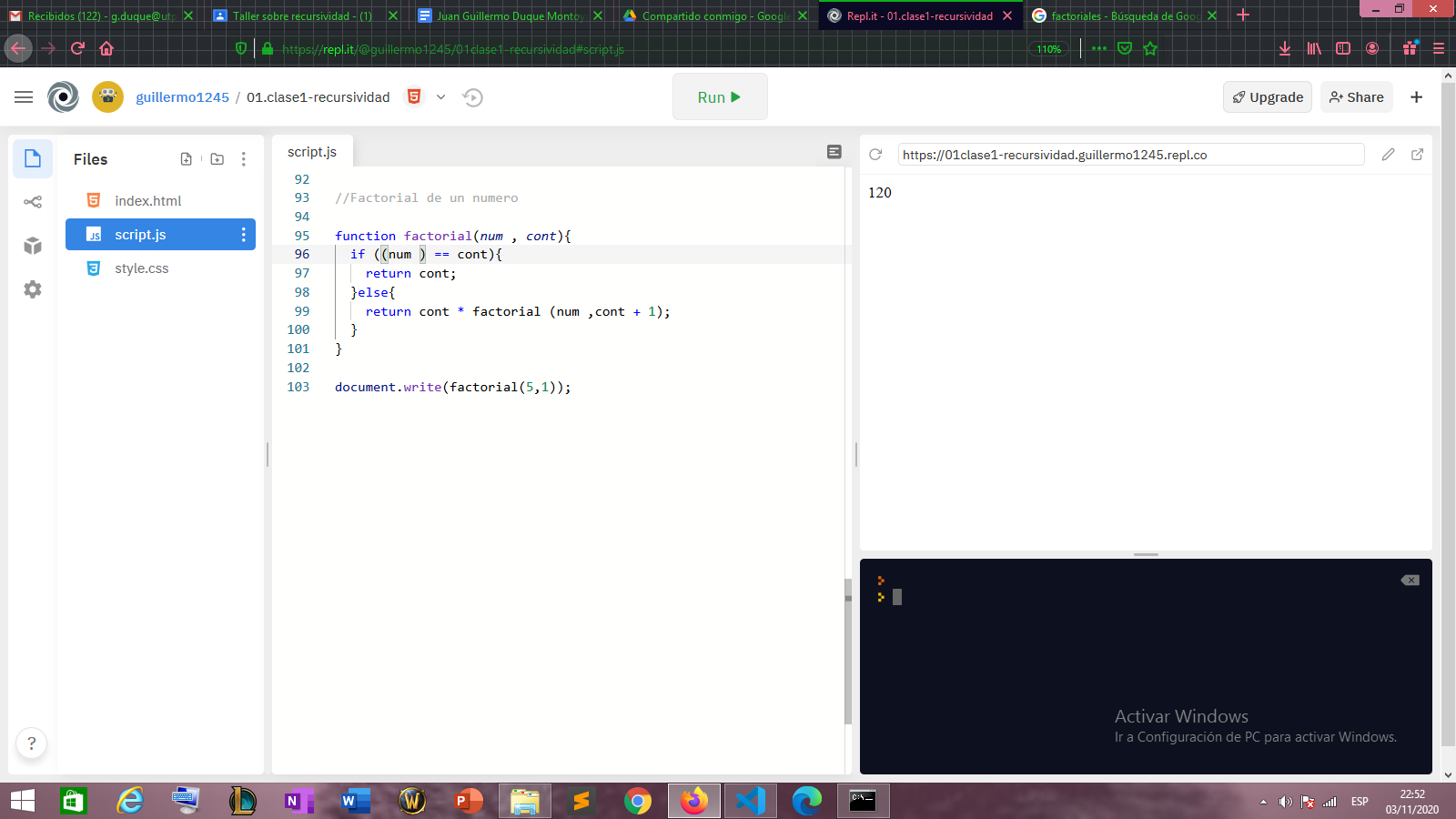


Figura 27.Recursividad 09

La solución que le di al primer ejercicio consiste en agregar un contador como parámetro a la función, de esta manera podía ir multiplicando el contador, y por medio de la recursividad lo iba sumando,

así:

cont = 1

cont = 2

cont= 3

cont =4

cont = 5

y el cont se irá sumando hasta que sea igual al número ingresado,cuando esto suceda se multiplicaran todos los cont obtenidos.

así:

1 \* 2 \* 3 \* 4 \* 5 == 120.

solución en :

https://repl.it/@guillermo1245/01clase1-recursividad#script.js

1. sumatoria (1, n) = n + sumatoria (1, n - 1 )

SOLUCION

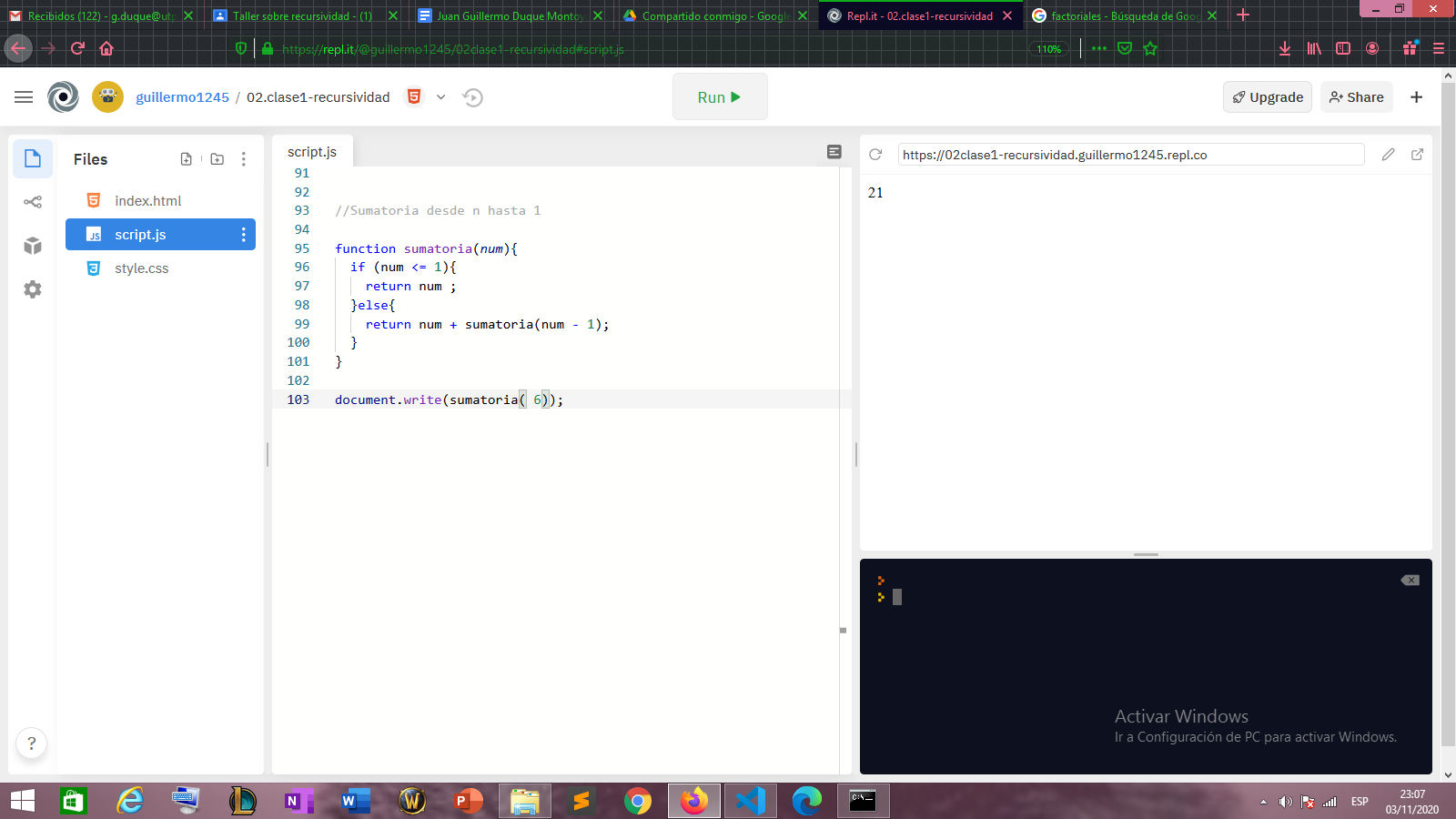
****

Figura 28. Recursividad 10

la solución que le di a este problema es un poco más sencilla, ya que lo que hice fue crear una condición donde se evalúa si el número es menor o igual a 1, si es afirmativa se imprime el mismo número, pero si la condición no es verdadera, se pasa al lado falso el cual llama a la recursividad y almacena una suma del argumento num.

algo así:

si num <= 1 retorna 1

pero si no

se almacena

num + num + num + num

 y en cada uno se le resta 1, hasta que la condición de arriba sea verdadera, cuando esto ocurra se retornara toda la suma acumulada hasta el momento.

la solución se encuentra aquí:

https://repl.it/@guillermo1245/02clase1-recursividad#script.js

1. Calcular x elevado a la y. Ejemplo: 5 elevado a la 3.

SOLUCION

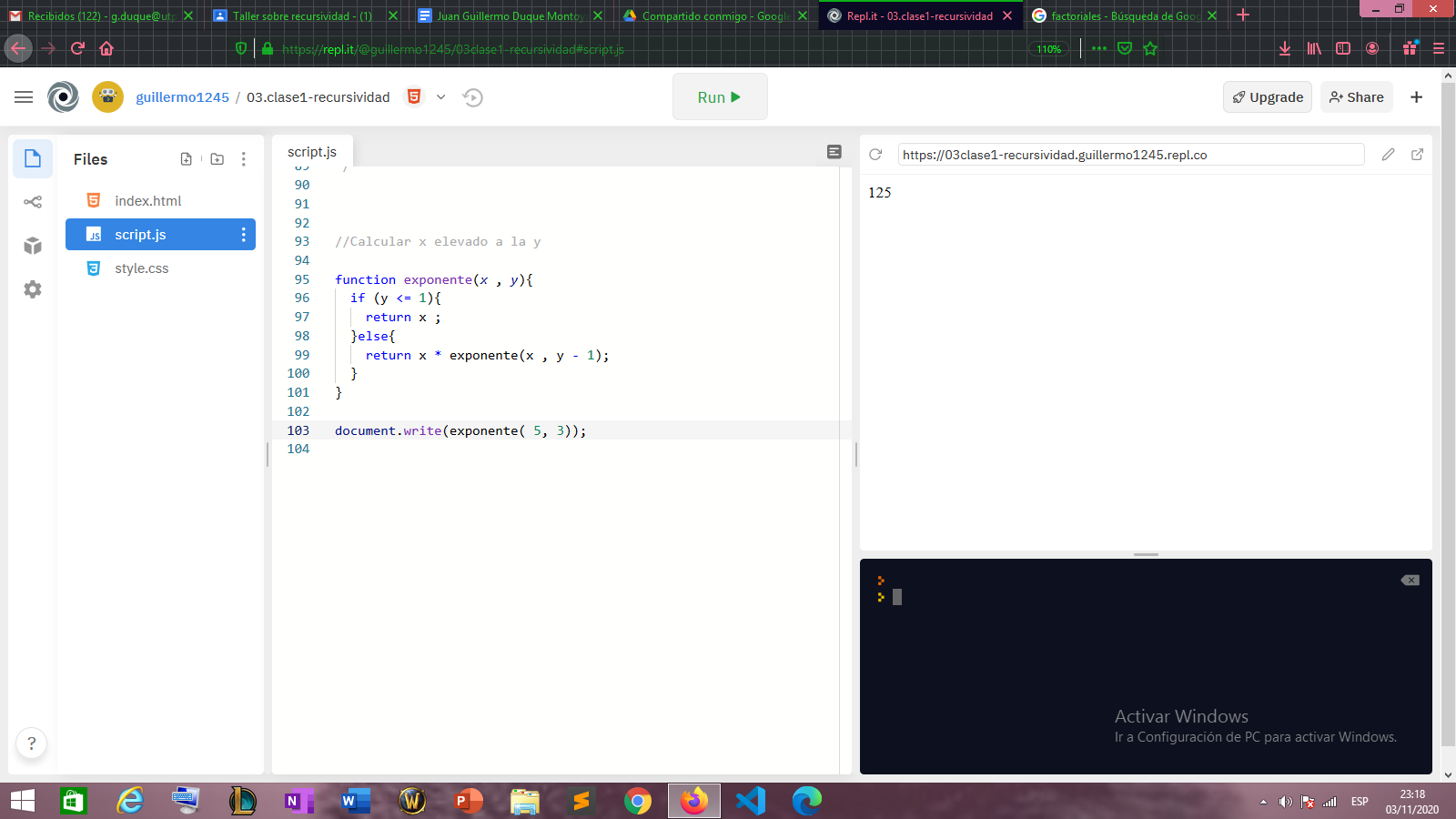
****

Figura 29. Recursividad 11

Para solucionar este ejercicio cree una función y le di dos argumentos X y Y en donde Y es el exponente de X, para darle recursividad a la función se le restaba a y hasta que fuera igual o menor a 1, esto con el fin de que las veces que se restara Y iban a ser el mismo número de veces que x se multiplicaba con x

algo así:

**fase1**

Y= 3

X = 5

5

y = Y- 1

**Fase 2**

y= 2

X= 5

5 \* 5

Y = Y - 1

**Fase 3**

**Y= 1**

**X= 5**

**5 \* 5 \* 5**

**Y=-1**

**Fase 4**

Y= 0

aquí como y es menor que 1 se termina la función y se retorna la multiplicación que en este caso es 125

La solución se encuentra aquí:

<https://repl.it/@guillermo1245/03clase1-recursividad#script.js>

Con lo visto hasta ahora es muy probable que hayamos entendido una totalidad el tema de la recursividad, sin embargo, veremos algunos ejemplos más, para reforzar el tema, y posterior nuevas preguntas, para practicar.

**EJEMPLO RECURSIBIDAD FACTORIAL**

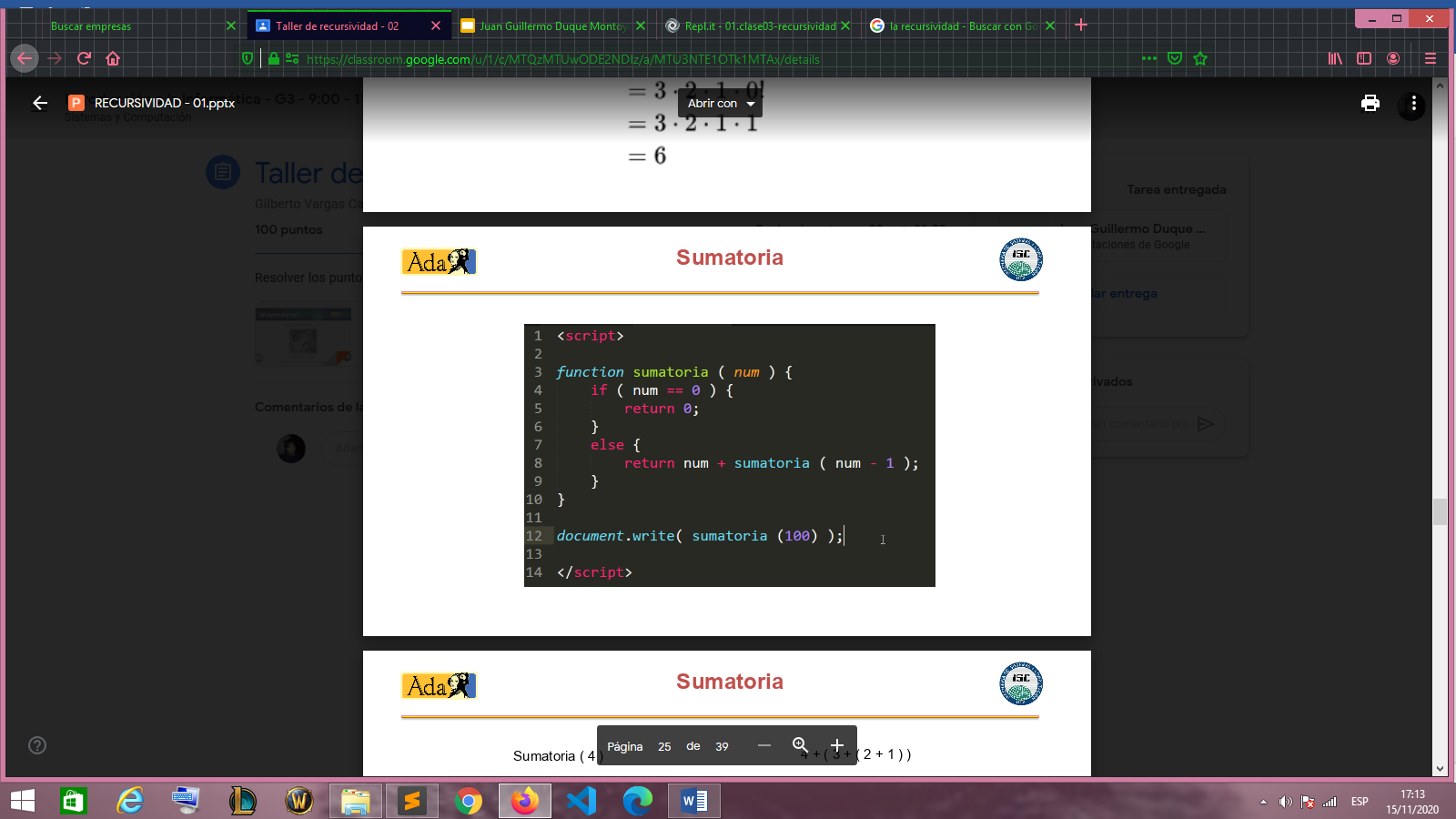
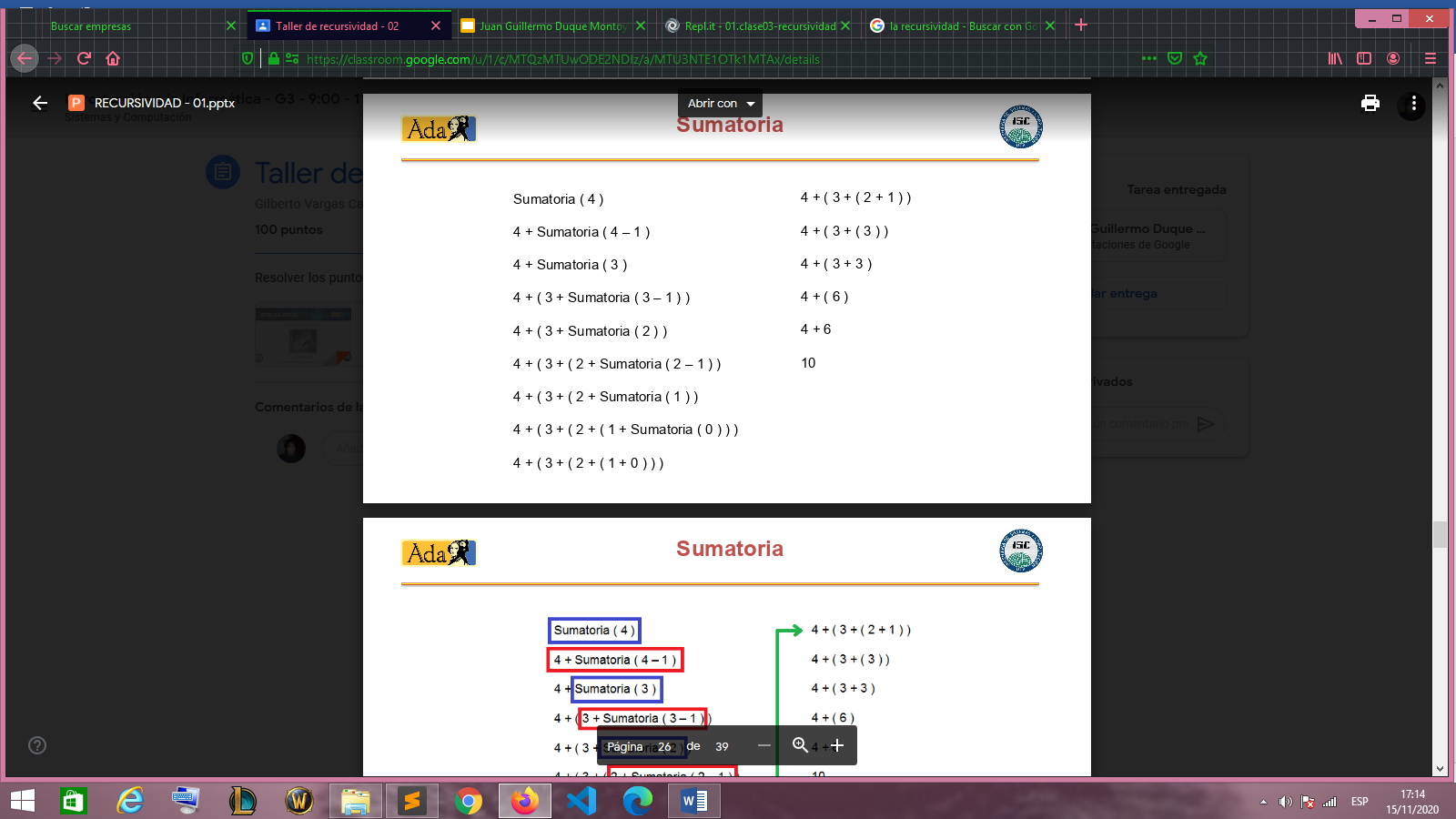


Figura 30. Recursividad 12

SOLUCION PASO A PASO:



**EJEMPLO ITERACION SUMAR ELEMENTOS DE UNA LISTA**

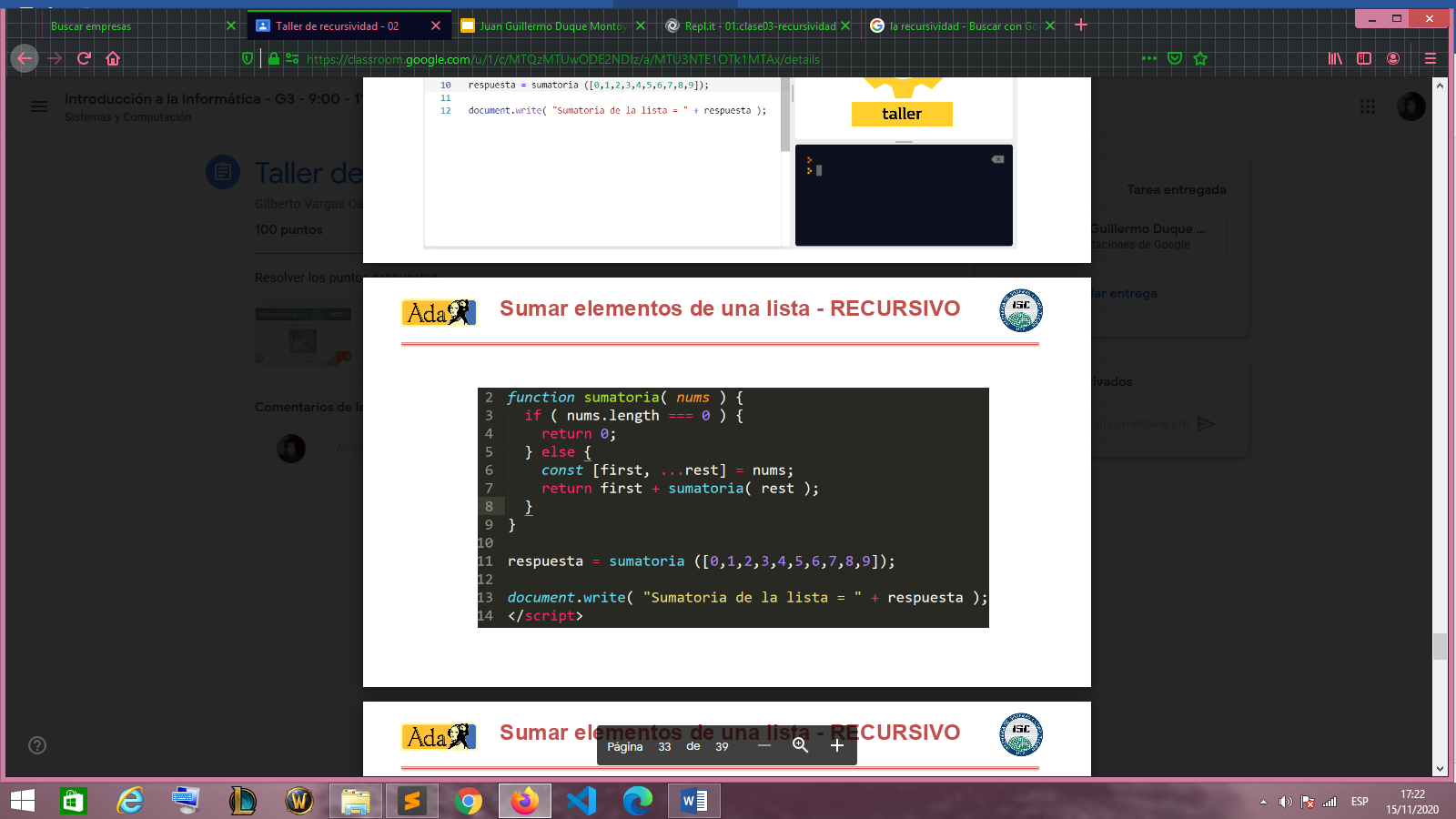


Figura 31. Recursividad 13

AL ejecutar el código, obtenemos como resultado:

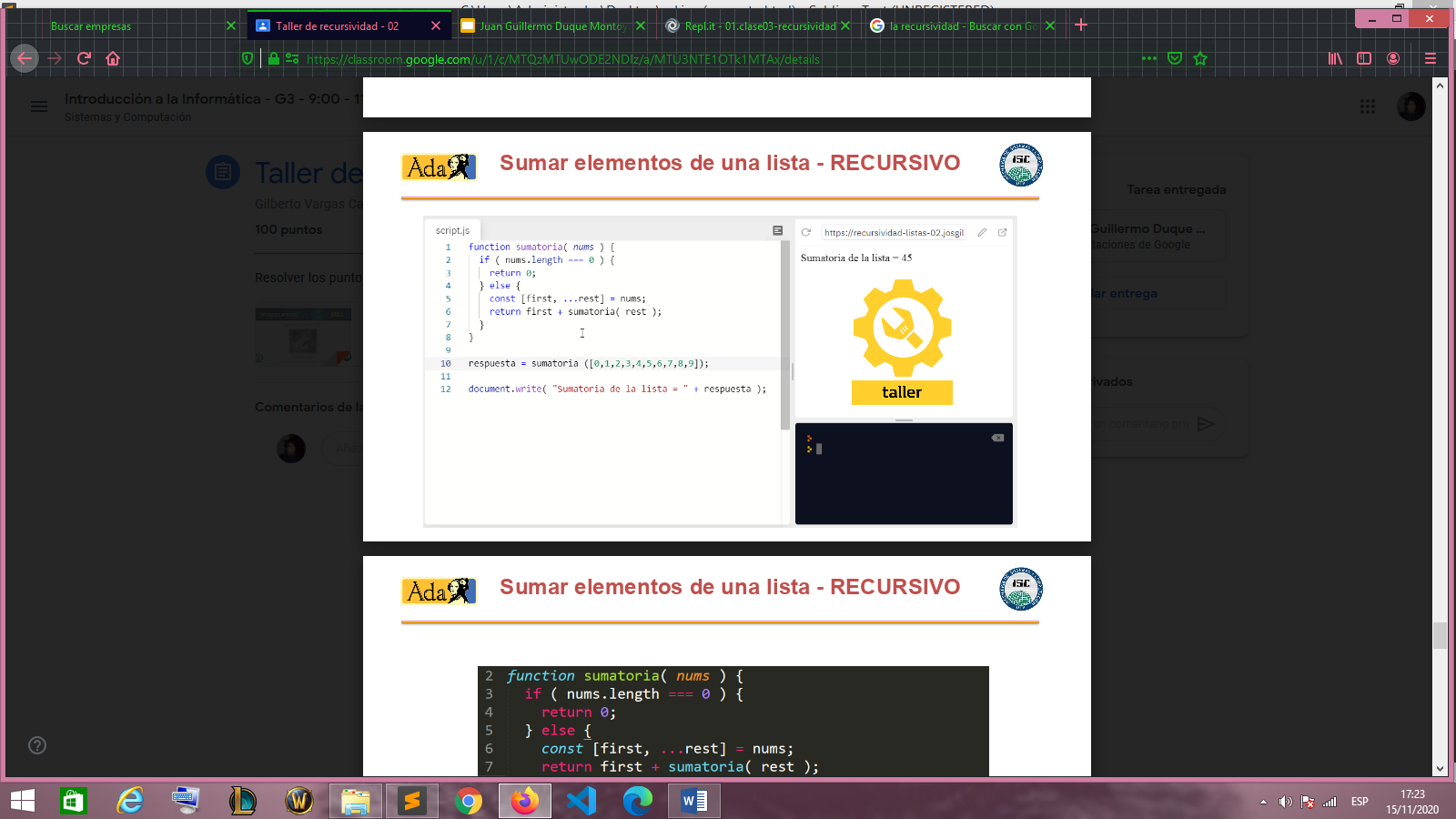


Figura 32. Recursividad 14

Ahora con los ejemplos planteados, podremos ver 3 nuevas preguntas que nos ayudaran a finalizar de manera exitosa el último tema de esta monografía que es la recursividad.

TALLER

1. Encuentra el elemento mayor en el

Lista

SOLUCION

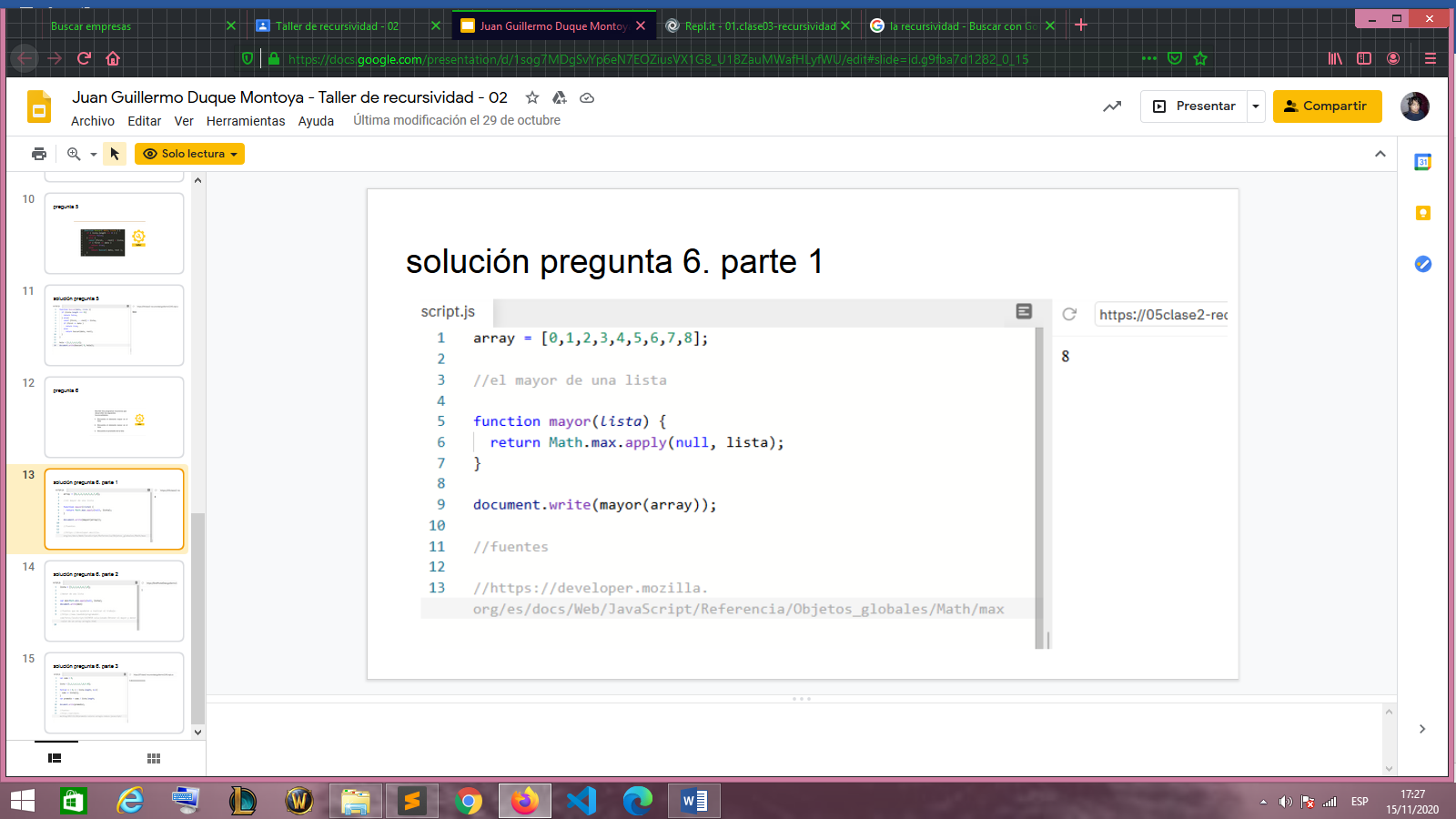


Figura 33. Recursividad 15

1. Encuentra el elemento menor en el

Lista

SOLUCION:

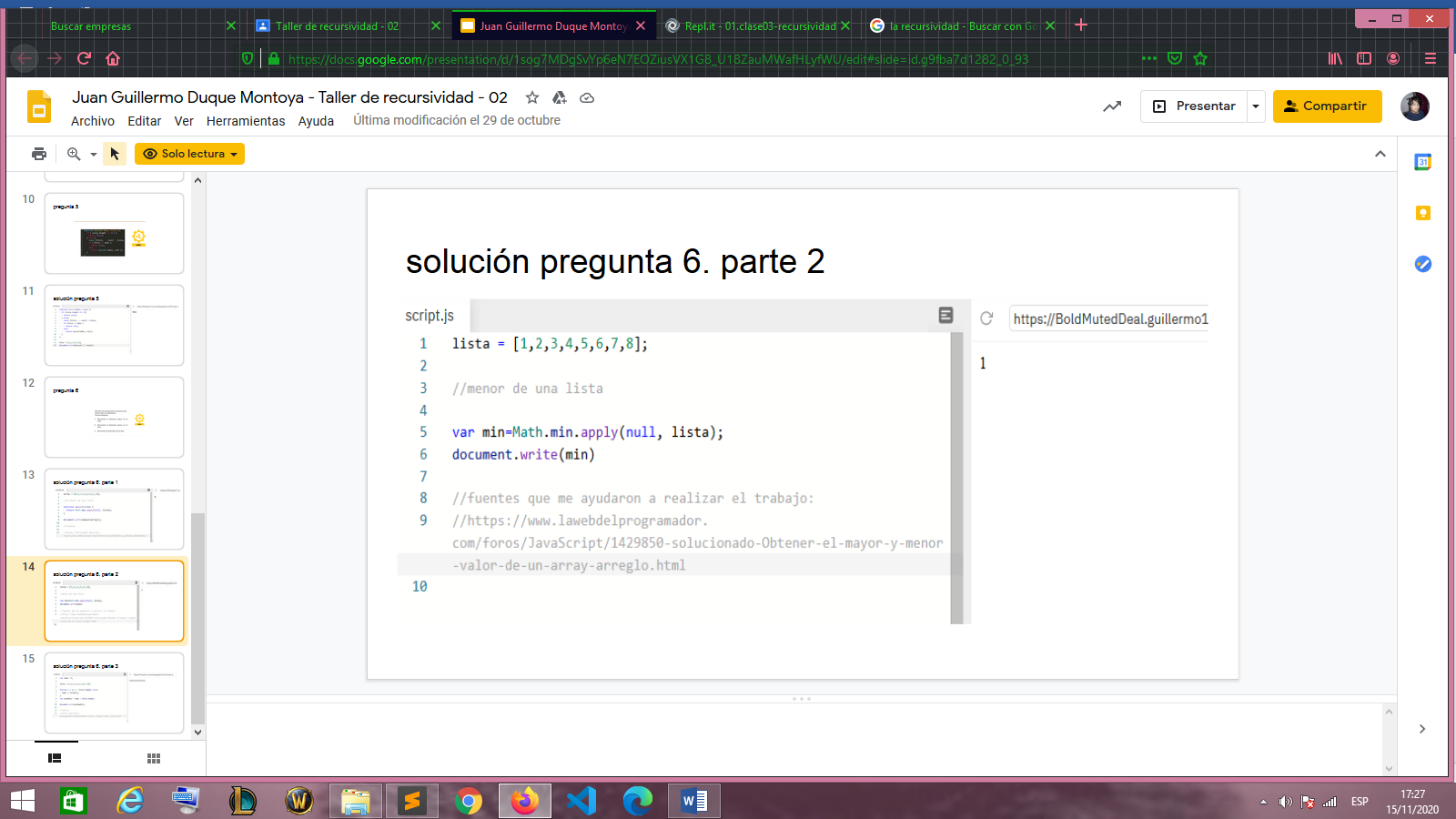


Figura 34. Recursividad 16

1. Encuentra el promedio de la lista

SOLUCION



Figura 35. Recursividad 17

# CONCLUSIONES

El desarrollo de las temáticas elaboradas en clase utilizando el lenguaje JavaScript prueba ser un mecanismo de gran valor para el aprendizaje de los conceptos básicos de la materia, esto se debe a las acciones que se deben realizar para crear este tipo de documentos, peor a modo de conclusión no desde el punto de materia o clase, si no desde las perspectivas personales de los temas vistos, puedo decir que los números binarios son muy útiles y extensos para muchos aprendizajes, y adicional a esto a la hora de mencionar la recursividad y todos los temas que este tema abarca, podemos decir que es una herramienta fundamental para todo programador, y que su utilidad es casi que infinita.

# BIBLIOGRAFÍA

1. <https://repl.it>
2. https://www.lawebdelprogramador.com/foros/JavaScript/1429850-solucionado-Obtener-el-mayor-y-menor-valor-de-un-array-arreglo.html
3. <https://www.definicionabc.com/comunicacion/recursividad.php>
4. <https://parzibyte.me/blog/2017/11/20/promedio-valores-arreglo-reduce-javascript/>
5. <https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Math/max>
6. https://ukode.es/que-es-y-para-que-sirve-la-recursividad/