# UNIDAD 1

# 

# INTRODUCCION

# 1. Que es la Programación de computadoras?

# La programación es el proceso utilizado para idear y ordenar las acciones necesarias para realizar un proyecto, preparar ciertas máquinas o aparatos para que empiecen a funcionar en el momento y en la forma deseados o elaborar programas para su empleo en computadoras.

# 2. Que es un lenguaje de Programación?

# Un lenguaje de programación es un lenguaje formal (o artificial, es decir, un lenguaje con reglas gramaticales bien definidas) que le proporciona a una persona, en este caso el programador, la capacidad de escribir (o programar) una serie de instrucciones o secuencias de órdenes en forma de algoritmos con el fin de controlar el comportamiento físico o lógico de una computadora, de manera que se puedan obtener diversas clases de datos o ejecutar determinadas tareas. A todo este conjunto de órdenes escritas mediante un lenguaje de programación se le denomina programa.

# 3. Lenguajes más usados en el mundo de la programación actual:

# 3.1. Python:

# Python destaca por ser un lenguaje de programación fácil de aprender y de usar. Es de código abierto y se usa en muchos campos como el Machine Learning, la automatización y la computación científica. Ha ganado tanta popularidad que incluso ha sobrepasado a Java en el ranking de GitHub en 2019.

# 3.2. Java:

# Java es especialmente popular para aplicaciones web servidor-cliente. Desde 2007, Sun Microsystems (ahora parte de Oracle) ha vuelto a licenciar la mayoría de tecnologías de Java bajo Licencia Pública General GNU.

# 3.3. *JavaScript:*

# *JavaScript (JS) es un indispensable entre los desarrolladores de software y un punto de partida estupendo para los principiantes. Junto con HTML y CSS, es una tecnología base de la World Wide Web. Es fácil de aprender Javascript, si lo practicas adecuadamente podrás entender más fácilmente otros lenguajes de programación, es el mejor lenguaje aplicado a la web.*

# *Javascript es un lenguaje de programación, o lo que es lo mismo, un mecanismo con el que podemos decirle a nuestro navegador que tareas debe realizar, en qué orden y cuantas veces*

# 3.4. C#:

# C# (C Sharp) se usa para el desarrollo de servicios web, juegos y aplicaciones web, móviles y de escritorio. Fue desarrollada por Microsoft en el 2000.

# 3.5. PHP:

# PHP se usa de manera generalizada para el desarrollo web en el lado del servidor y para crear contenido dinámico de páginas web. Sin embargo, también se puede usar para muchas otras tareas de programación como el control robótico de drones.

# 3.6. C/C++:

# C destaca por la eficiencia de su código. Es muy popular para desarrollar software de sistemas; y también se usa para crear aplicaciones. C++ fue desarrollado como una extensión de C. Se usa en infraestructura de software, aplicaciones de escritorio y servidores, entre otros.

# 3.7. TypeScript:

# TypeScript está diseñado para desarrollar aplicaciones de gran tamaño, aplicaciones JavaScript para para ejecutar en el lado del cliente y del servidor, etc. Es un lenguaje de programación de código abierto desarrollado por Microsoft.

# 4. Que usar para programar en Javascript?

# Entre los ambientes de desarrollo para programar en Python, encontramos, a la consola del lenguaje, los editores de código y los IDE.

# Entre los editores de código más usados, encontramos los siguientes:

# 7.1. Visual Studio Code:

# 7.2. Sublime Text:

# 7.3. Brackets:

# PRIMEROS PASOS CON JAVASCRIPT

# ¿Qué es JavaScript?

****Java Script es un lenguaje interpretado (no requiere de un compilador para ser ejecutado) que se utiliza tradicionalmente en páginas web dinámicas; este lenguaje nos permite incorporar interactividad de las páginas con el usuario de las mismas; por ejemplo, texto que desaparece, animaciones, acciones que se activan al dar clic sobre un botón, etc.

Muchas de las tareas que realizamos con HTML y CSS se podrían realizar con Javascript. De hecho, es muy probable que al principio nos parezca que es mucho más complicado hacerlo con Javascript, y que por lo tanto no merece la pena. Sin embargo, con el tiempo veremos que Javascript nos ofrece una mayor flexibilidad y un abanico de posibilidades más grande, y que bien usadas, pueden ahorrarnos bastante tiempo.

Se debe aclarar que este lenguaje de scripts nada tiene que ver con el lenguaje JAVA y no se debe confundir con él; tienen alginas similitudes, pero no son lo mismo y ninguno depende del otro.

Para programar código Se utiliza entre las etiquetas **<script></script>**

## **Como se visualiza JavaScript**

JavaScript tiene diferentes formas de salida de datos, así:

* Escriba en un elemento HTML contenido diverso, utilice **innerHTML**.
* Muestre en el navegador web, contenido HTML utilice **document.write().**
* Muestre un mensaje en un cuadro de dialogo tipo alerta, utilice **window.alert().**
* Escriba un mensaje en la consola del navegador, use **console.log().**

## Hablemos de la SINTAXIS

En JavaScript como en los demás lenguajes de programación, se deben seguir ciertas reglas para escribir los comandos o líneas de programación. Aquí describimos algunas de estas reglas que permitirán realizar Scripts bien estructurados y guardando o respetando las características técnicas del lenguaje:

1. No se tienen en cuenta los espacios en blanco ni los saltos de línea y esto nos permite tabular las líneas de código para establecer jerarquías y de esta manera comprenderlo mejor.
2. Es sensible a MAYUSCULAS y minúsculas, es decir que para que funcione adecuadamente un Script debemos utilizar de manera adecuada tanto mayúsculas como

minúsculas, sobre todo al definir variables u objetos sobre los cuales se creara código de programación que los pueda llegar a afectar.

1. No se requiere establecer el tipo de dato que se almacenara en las variables y esto nos permite definir una variable y guardar en ella diferentes tipos de datos durante la ejecución de un Script sin que JavaScript reporte errores por este hecho.
2. Aunque se recomienda terminar cada línea de programación con un punto y coma (;), en JavaScript no es necesario ni se obliga.
3. Se pueden utilizar comentarios para documentar un programa y además se recomienda el uso de los mismos para dar legibilidad y claridad a los programas escritos. Esta es una buena costumbre que recomiendo no abandonar. Para comentar una sola línea se utiliza // al principio de la línea y para comentar varias líneas utilizamos /\* al inicio y \*/ al final de la última línea de comentario.

## Estructuras básicas de programación en JavaScript

## Variables y constantes

Una variable como todos sabemos, es un espacio de memoria cuyo contenido puede variar dentro o después de la ejecución de un programa; al nombre que se le da a dicha variable también se le puede denominar identificador. Es importante destacar que las variables hacen más recursivos y eficientes los programas y que los identificadores en todo lenguaje de programación se rigen por ciertas reglas que recordaremos a continuación:

1. No deben contener caracteres especiales ni espacios en blanco y no deben empezar por un valor numérico.
2. Debe ser lo suficientemente significativo que permita de manera lógica determinar qué tipo de dato se almacenara en la variable.
3. No debe ser ni muy extenso ni muy corto; debe tener una longitud razonable que permita su fácil comprensión.
4. El único carácter especial que se permite para un identificador es el guion bajo, raya baja o raya al piso (\_).
5. JavaScript distingue en los identificadores entre mayúsculas y minúsculas.

Normalmente las variables en JavaScript se definen utilizando la palabra reservada var tal como se ve en los siguientes ejemplos:

**var numero1=3;**

**var numero2=5;**

**var resultado=numero1+numero2;**

**const mes=1;**

Observemos que cada línea de programación termina con un punto y coma (;) situación que ya habíamos mencionado anteriormente. Se aclara que las variables solo son definidas una sola vez, es decir que la palabra reservada var solo se utiliza cuando se definen por primera vez las variables, luego no es necesario utilizarla. Pero recordemos que JavaScript es un lenguaje que no obliga a definir variables antes de utilizarlas, ya que se pueden utilizar en el momento que se requieran y aunque esta es una práctica que no se recomienda, tratando de respetar los buenos hábitos o las buenas prácticas en programación.

## Tipos de variables y constantes

**Numéricas:** nos permiten guardar información de tipo numérico (Enteros, decimales, etc).

**const iva=19;**

**var Estatura=1.85;**

Como podemos observar los valores decimales utilizan como separador el punto (.) y no la coma (,) en JavaScript.

**Cadena de caracteres:** utilizadas para almacenar texto, caracteres, palabras, etc. Para asignar un valor textual a una variable este se debe encerrar entra comillas dobles (“ “) **o** simples (‘ ‘) según el caso.

**var mensaje=” Bienvenidos al INCAP”**

**var Nombre\_Producto= “Disco Duro”**

**alert( mensaje + "\n" + Nombre\_Producto );**

**document.write(mensaje + "<br>" + Nombre\_Producto );**

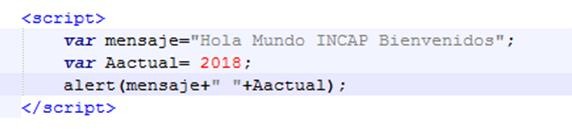
Cuando se requiere de incluir caracteres especiales que generen algún tipo de problemas es necesario utilizar unas combinaciones de caracteres que a continuación se describen:

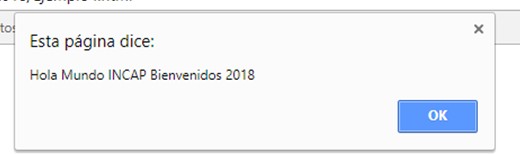
|  |  |
| --- | --- |
| **Incluir** | **Combinación de caracteres** |
| Una Nueva línea | \n |
| Un tabulador | \t |
| Una comilla simple | \’ |
| Una comilla doble | \” |
| Una barra inclinada | \\ |

Veamos un ejemplo de variables inicializadas con datos de tipo cadena de caracteres y numéricos:

# 

# Codigo JavaScript



**Resultado**

* + 1. **Arrays**

También llamados vectores, matrices o arreglos, me permiten agrupar en una salo variable una colección de elementos que pueden ser del mismo tipo o de tipos diferentes.

Supongamos que queremos realizar algún tipo de procedimiento con los días de la semana; para ello podríamos crear siete variables para almacenar cada día de la semana y trabajar con cada uno de ellos; pero si en lugar de crear siete variables se crea una única variable que me permita guardar en ella los siete días de la semana y poder trabajar con cada uno de ellos independientemente, esto nos permitiría crear un código de programación más limpio, eficiente y dinámico.

Para declarar una variable tipo array, se utilizan los caracteres […] de la siguiente manera:

**Var dias\_semana=[“Lunes”,“Martes”,”Miercoles”,”Jueves”,”Viernes”,”Sabado”,”Domingo”]**

Ahora para acceder a cada elemento solo basta con indicar el índice o la posición del mismo, teniendo en cuenta que los índices o posiciones en programación casi siempre empiezan en 0. Así, entonces si quisiera acceder o mostrar el primer día de la semana que es ***lunes***, se podría utilizar el siguiente código:

En este ejemplo accedemos a la posición 0 del array que será el día lunes

**var PrimerDia=dias\_semana[0];**

**alert(PrimerDia);**

* + 1. **Objetos en JS**

Los objetos en Javascript son variables que pueden almacenar datos de diferentes características.

**var producto = {categoria:**"Silla"**, marca:**"rimax"**, color:**"blanca"**};**

Para acceder a las propiedades del objeto producto se haría de la siguiente forma:

*Nombreobjeto.nombrepropiedad como p. ejemplo: producto.marca y producto.color*

* + 1. **Operadores:**

Los operadores permiten la manipulación de los valores almacenados en las variables y así verdaderamente sacarle el mayor provecho a la programación; por eso a continuación describimos de manera básica cada una de las clases de operadores:

|  |  |  |
| --- | --- | --- |
| **De Asignación** | | |
| **=** | Me permite crear un incremento fijo de uno en uno a una variable | **var contador=contador + 1;** |
| **+=** | Permite crear un acumulador de valores numéricos | **Tot += edad;** |
| **De Incremento y Decremento** | | |
| **++** | Incrementa en uno el valor de una variable | **var Valor1=5;**  **++Valor1;**  **//Valor1 es ahora 6** |
| **--** | Decrementa en uno el valor de una variable | **var Valor1=5;**  **--Valor1;**  **//Valor1 es ahora 4** |
| **Lógicos** | | |
| ! (Negación) | Se utiliza para obtener el valor contrario de la variable. | **var Cantidad=0;**  **!Cantidad;**  **// Cantidad=true Var Cantidad=2;**  **!Cantidad;**  **//Cantidad=false** |
| && (AND) | Está relacionado con la tabla de verdad And y obtiene su resultado de la combinación de  dos valores booleanos. | **var Valor1=true; var Valor2=true;**  **Resultado=Valor1&&Valor2;**  **//Resultado es true** |
| || (OR) | Está relacionado con la tabla de verdad Or y obtiene su resultado de la combinación de  dos valores booleanos. | **var Valor1=true; var Valor2=false;**  **Resultado=Valor1||Valor2;**  **//Resultado es true** |
| **Matemáticos** | | |
| + , - , \* , / ,  %(Modulo) | Me permiten generar las diferentes operaciones matemáticas básicas. Una forma de simplificar las expresiones matemáticas es utilizar el operador antes del  signo igual: +=, -=, \*=, /= | **var Valor1=5; var Valor2=2; Mult=6;**  **Suma=Valor1+Valor2; Resta=Valor1-Valor2; Residuo=Valor1%Valor2;**  **Mult\*=4;** |
| **De Relación** | | |

|  |  |  |
| --- | --- | --- |
| > (Mayor que)  < (Menor que)  >= (Mayor o igual)  <= (Menor o igual)  == (Igual, comparación)  != (Diferente) | Estos operadores nos permiten comparar dos valores y su resultado siempre será un resultado lógico el cual puede ser falso o verdadero, según el caso.  Es importante anotar que cuando se comparan cadenas de caracteres, las que anteceden son menores a las que preceden y las mayúsculas son menores que las  minúsculas. | **Var Num1=2; Var Num2=6;**  **Comp=Num1>Num2;**  **// Comp quedara con false Comp2=Num1!=Num2;**  **// Comp2 quedara con true Comp3=Num1==Num2;**  **// Comp3 quedara con false Comp4=Num1<Num2;**  **// Comp4 quedara con true** |

* 1. **Comentarios en Javascript**

Los comentarios en JavaScript tienen dos utilidades:

* Para explicar el código JavaScript y así poder documentar y entenderlo más fácilmente.
* Para evitar la ejecución de un fragmento de codigo de forma eventual.
* Los comentarios se escriben comenzando por dos símbolos: **//**
  1. **Consola en Javascript**

La consola, es una pestaña que integra el inspector de elementos (pestaña ***element)*** que es una sección del navegador web donde tenemos control a varios aspectos referentes a la página actual visualizada como, por ejemplo: su estructura de etiquetas HTML, sus estilos CSS, sus scripts, entre otros, en la pestaña ***Console*** podemos visualizar los resultados de Javascript utilizando la sentencia *console.log,* también podemos ejecutar funciones o sentencias de Javascript.

**Atajos de teclado:**

1. Para ir al Inspector de elementos CTRL + SHIFT + I
2. Para ir a la pestaña consola: CTRL + SHIFT + J
3. Para borrar la consola: CTRL + L o tecla F12
   1. **Strings**

Los strings en JavaScript se declaran para almacenar cadenas de textos.

Ejemplo

Var marca=”huawei”;

**FDEH**

* Estudiante en el siguiente ejercicio debes de completar las siguientes definiciones:

Cuáles son las diferencias entre utilizar variables y contantes ?\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

* Las variables en Javascript necesitan definir su tipo de datos a utilizar sí o no y porque?\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# DOM (Document object Model)

Esta novedad en JavaScript les permite a los programadores acceder y manipular a su antojo las paginas HTML. Este lenguaje se ha convertido en una utilidad disponible para la gran mayoría de lenguajes de programación, entre ellos PHP, Java, JavaScript, etc.

Algunas de las tareas que se facilitan aplicando las ventajas de DOM son entre otras: obtener un valor almacenado por algunos elementos de un formulario; crear un elemento (párrafo <p>,

<div>, etc.) de manera dinámica y poderlo añadir a una página; añadir movimiento o animación a un elemento o hacer que aparezca o desaparezca. Para poder sacarle todo el provecho a DOM se requiere realizar una trasformación en la página HTML ya que esta es una sucesión de caracteres, característica que hace difícil su manipulación; esta trasformación la mayoría de navegadores la realizan de manera automática aplicándoles una estructura más eficiente a las paginas HTML para manipular.

DOM transforma todos los elementos de una página HTML en un conjunto de elementos llamado Nodos los cuales se encuentran interconectados formando un árbol de Nodos, y al igual que un árbol vamos a encontrar ramificaciones determinando de esta manera jerarquías.

Miremos un ejemplo:

<**html**>

<**head**>

*<title>*HTML Ejemplo</title>

<*meta* charset="UTF-8">

**</head>**

<**body** >

En esta estructura básica de DOM HTML vemos que el nodo principal <HTML> luego siguen los nodos <head> y <body> y después de esos nodos vendrán los subnodos respectivos. Ver imagen de árbol de nodos.

<*h1*>DOM HTML</h1>

*<p>*

Es el árbol de objetos de HTML

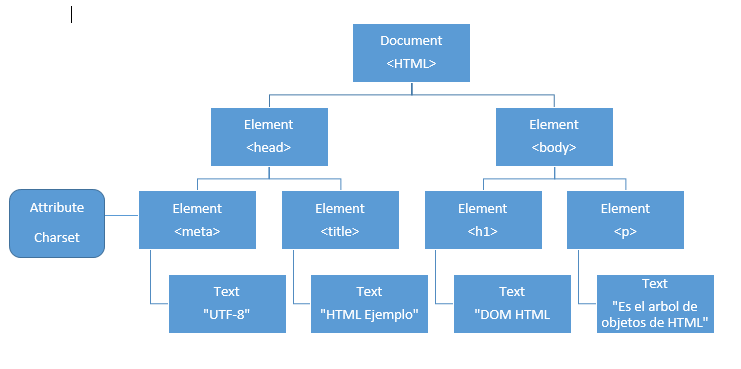
</p>

**</body>**

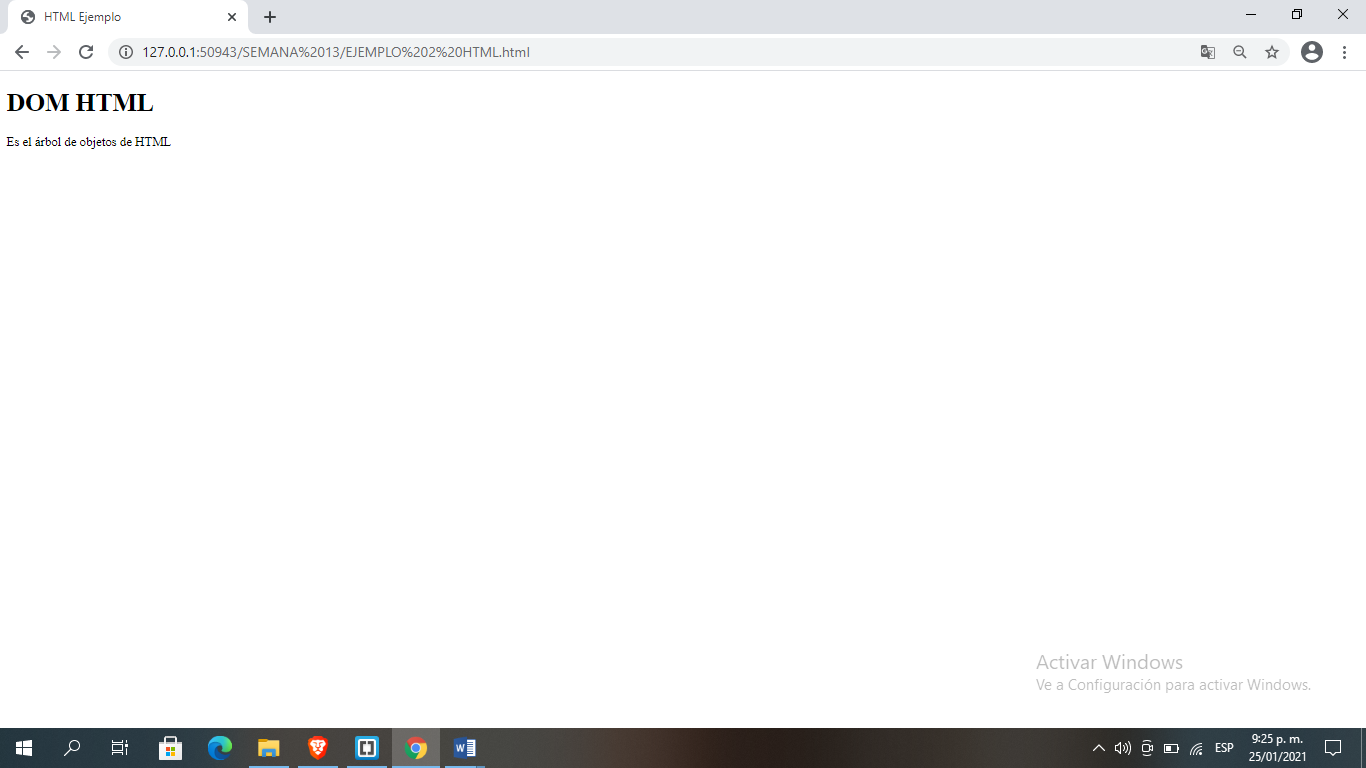
**</html>**

Miremos un ejemplo:

La página anterior la podemos trasladarla al siguiente árbol de Nodos:



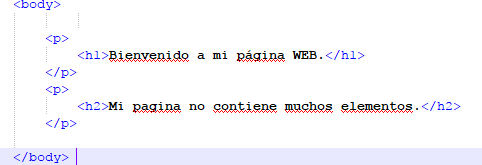
**RESULTADO FINAL:**



Aunque el ejemplo maneja una página HTML muy sencilla, esta se utilizó para entender de manera igualmente sencilla el concepto del DOM.

La transformación automática de una página HTML en un árbol de nodos siempre sigue las mismas reglas:

1. Las etiquetas HTML se transforman en dos nodos: el primero corresponde a la propia etiqueta y el segundo nodo es hijo del primero y normalmente corresponde al texto contenido en la primera etiqueta o primer nodo.
2. Si una etiqueta HTML se encuentra dentro de otra, se realiza el mismo proceso anterior, con la diferencia que los nodos resultantes serán hijos de su etiqueta padre.

Un ejemplo delo anterior sería:

Y este sería el árbol de Nodos:



## Tipos de Nodos

Aunque podemos encontrar 12 tipos de nodos, mencionaremos los cinco principales mas no los únicos y recalco lo anterior mas no los únicos:

1. **Document**: nodo raíz del cual se desprenden todos los demás.
2. **Element**: representa cada una de las etiquetas HTML, este nodo es el único que puede contener atributos y del cual se desprenden otros nodos.
3. **Attr**: se define para representar cada uno de los atributos de las etiquetas HTML.
4. **Text**: nodo contenedor del texto definido en una etiqueta.
5. **Comment**: representa los comentarios incluidos en una página HTML.
   1. **OBJETO DOCUMENT –DOM-**

El objeto document es el propietario de todos los demás objetos de su página web, El objeto document representa toda la página web. Si desea acceder a un determinado elemento ubicado en una página HTML, siempre comience por colocar al objeto document.

***Ejemplo 1:***

<html>

<body>

<title>EJEMPLO 1 JS</title>

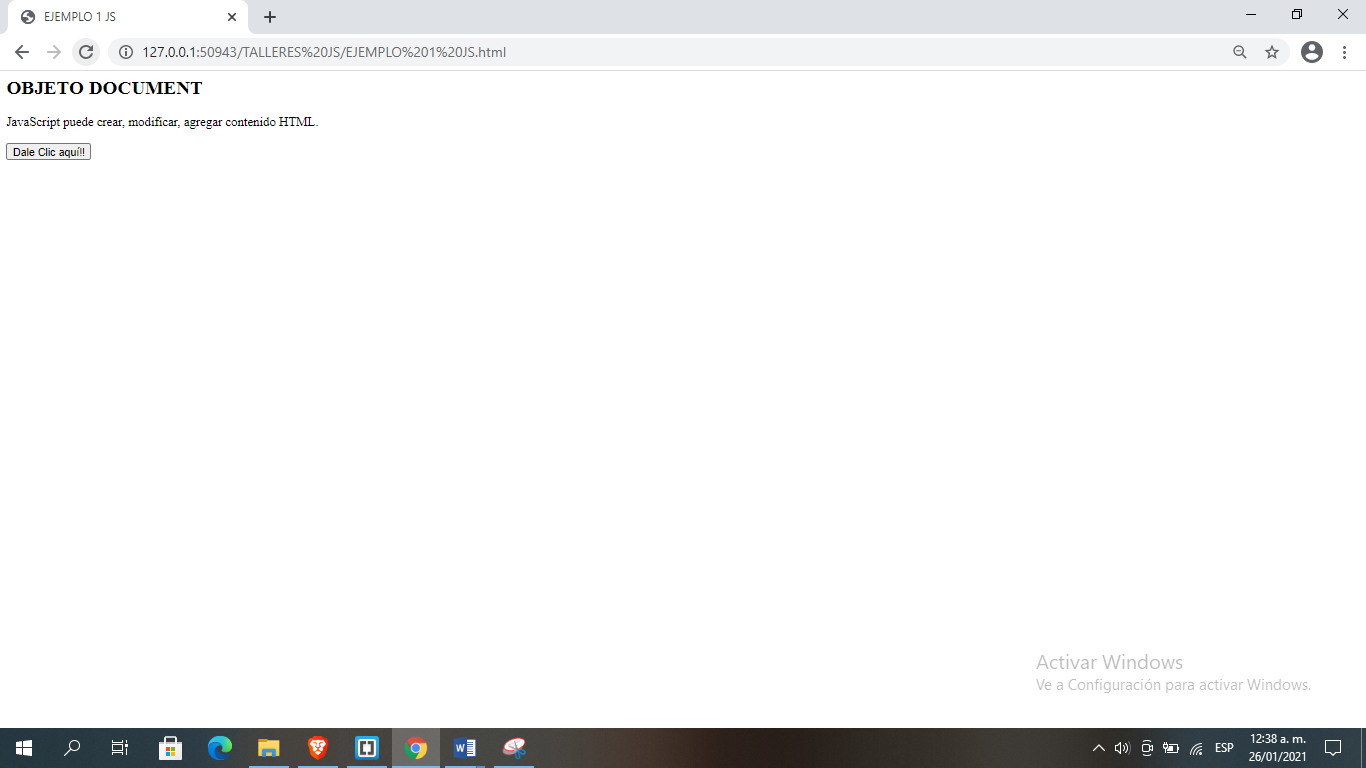
<h2>OBJETO DOCUMENT</h2>

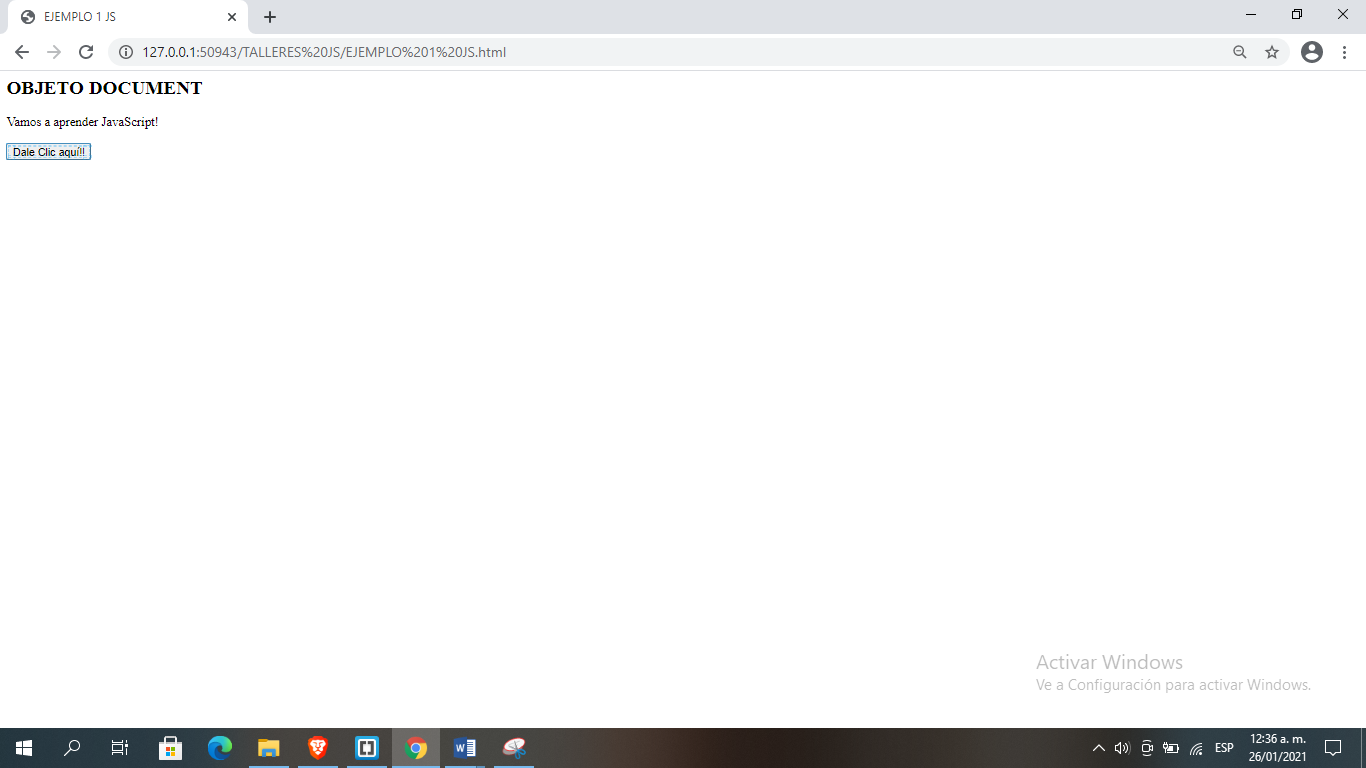
<p id="docto">JavaScript puede crear, modificar, agregar contenido HTML.</p>

<button type="button" **onclick**=**'document.getElementById("docto").innerHTML = "Vamos a aprender JavaScript!"'**> Dale Clic aquí!!</button>

</body>

</html>





***Ejemplo 2***

<html>

<body>

<title>EJEMPLO 1 JS</title>

<h2>OBJETO DOCUMENT</h2>

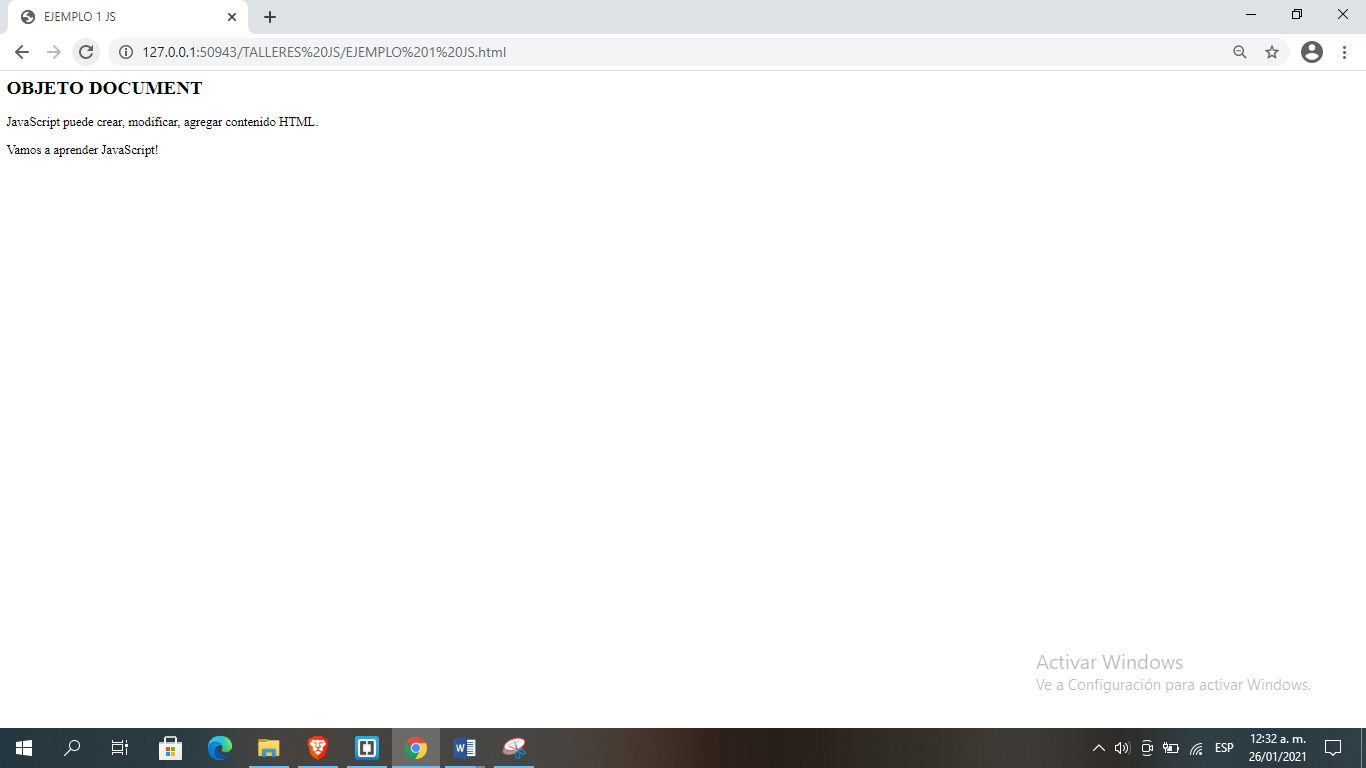
<p id="docto">JavaScript puede crear, modificar, agregar contenido HTML.</p>

<script>document.write("Vamos a aprender JavaScript!");

</script>

</body>

</html>



**FDEH**

Estudiante para reafirmar este conocimiento debes de realizar el árbol de nodos de este documento HTML

<html>

<head>

<title>PAGINA WEB INICIAL</title>

<meta charset="UTF-8">

</head>

<body bgcolor="lightblue">

<p><font color="red">PHP - JAVA - JAVASCRIPT- HTML- C++</font></p>

</body>

</html>

## Forma de acceder a los nodos

DOM permite acceder de dos formas diferentes a un nodo: el acceso a través de su nodo padre o de manera directa.

La primera forma es poco recomendable y es por esta razón que nos dedicaremos a explicar las formas en las que puedo acceder directamente a los diferentes nodos, sin necesidad de hacer todo el recorrido desde el nodo raíz, hasta el nodo buscado. Cabe anotar que el árbol de nodos es construido en su totalidad solo cuando la página HTML es cargada por completo y solo en este momento se pueden realizar las modificaciones pertinentes a los nodos respectivos.

* + 1. ***Encontrar elementos html***

|  |  |
| --- | --- |
| **Metodo Javascript** | **Que hace** |
| document.getElementById(*id*) | Buscar un elemento por su identificador id |
| document.getElementsByTagName(*nombre*) | Busca elementos por su etiqueta, este método genera un array |
| document.getElementsByName(*nombre*) | Buscar elementos por su clase name, este método genera un array |

* + 1. ***Sustituir elementos html***

|  |  |
| --- | --- |
| **Metodo Javascript** | **Que hace** |
| *element*.innerHTML =  *Nuevo Contenido html* | Cambia el Contenido de un element HTML |
| *element*.*attribute = Nuevo valor* | Cambia el valor del atributo de un element HTML |
| *element*.style.*propiedad = Nuevo estilo* | Cambia el estilo de un elemento HTML |
| **Metodo Javascript** | **Que hace** |
| *element*.setAttribute*(atributo, valor)* | Cambia el valor del atributo de un element HTML |

* + 1. ***Añadir y quitar elementos***

|  |  |
| --- | --- |
| **Metodo Javascript** | **Que hace** |
| document.createElement(*elemento*) | Crear un elemento HTML |
| document.removeChild(*elemento*) | Eliminar un elemento HTML |
| document.appendChild(*elemento*) | Agregar un elemento HTML |
| document.replaceChild(*nuevo, anterior*) | Reemplaza un elemento HTML |
| document.write(*texto*) | Escribe en el navegador web texto indicado |

* + 1. ***Agregar manejador de funciones y eventos***

|  |  |
| --- | --- |
| **Metodo Javascript** | **Que hace** |
| document.getElementById(*id*).onclick = function(){***codigo***} | Crea código Html en instrucciones para programar un evento de tipo onclick |

* + 1. ***Encontrar elementos HTML por selectores CSS***

|  |  |
| --- | --- |
| **Metodo Javascript** | **Que hace** |
| document.querySelector("nombre clase") | Busca todos los estilos del primer elemento CSS que cumplan el criterio de coincidencia con un selector CSS clase especificado |
| document.querySelectorAll("elemento nombre clase") | Busca todos los elementos HTML que cumplan el criterio de coincidencia con un selector CSS aplicado (id, nombres de clase) |

## Document.getElementsByTagName():

Este método getElementsByTagName (Nombres de Etiquetas) se utiliza para buscar y referenciar a todos los elementos de la página HTML que coincidan con el nombre de la etiqueta establecido dentro del paréntesis; este nombre de etiqueta es el parámetro que se pasa en la función.

<html>

<body>

<title>EJEMPLO 1 JS</title>

<h2>OBJETO DOCUMENT</h2>

<p id="docto">JavaScript puede crear, modificar, agregar contenido HTML.</p>

<script>document.write("Vamos a aprender JavaScript!");

</script>

</body>

</html>

La siguiente línea de script hace referencia a todos los párrafos <p> de la página HTML:

**var parrafos= document. getElementsByTagName(“p”);**

En el DOM document es el nodo principal a partir del cual se realiza la búsqueda de los elementos. El valor devuelto por la función es un **array** con todos los nodos <p> que cumplen con la condición establecida en el parámetro que se pasa en la función. Dicho array es de nodos y no de texto, situación que obliga a procesar los valores del array de la siguiente manera:

**var parrafo=parrafos[0];**

**El código anterior nos devuelve el primer párrafo de la página HTML**

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

**var parrafo=parrafos[i];**

**}**

El código anterior recorre todos los párrafos contenidos en la página HTML y el resultado lo almacena en la variable parrafo.

**EJEMPLO 2**

Se almacena en la constante parrafos el array de elementos p, se declara la constante parrafo de tipo array, se genera un bucle for para leer la cantidad de los elementos <p> que están en documento HTML, luego muestra en navegador, los parrafos y el objeto parrafo html del array

<!DOCTYPE html>

<html>

<title>EJEMPLO 2 JS</title>

<body>

<h2>OBJETO DOCUMENT</h2>

<p> JavaScript puede crear, modificar, agregar contenido HTML.</p>

<p>En este caso JavaScript muestra la cantidad de los elementos p.</p>

**<script>**

**const parrafos= document. getElementsByTagName("p");**

**const parrafo= parrafos[0];**

**for (let i=0; i<parrafos.length; i++) {**

**const parrafo=parrafos[i];**

**parrafo.innerHTML+= parrafos[i];**

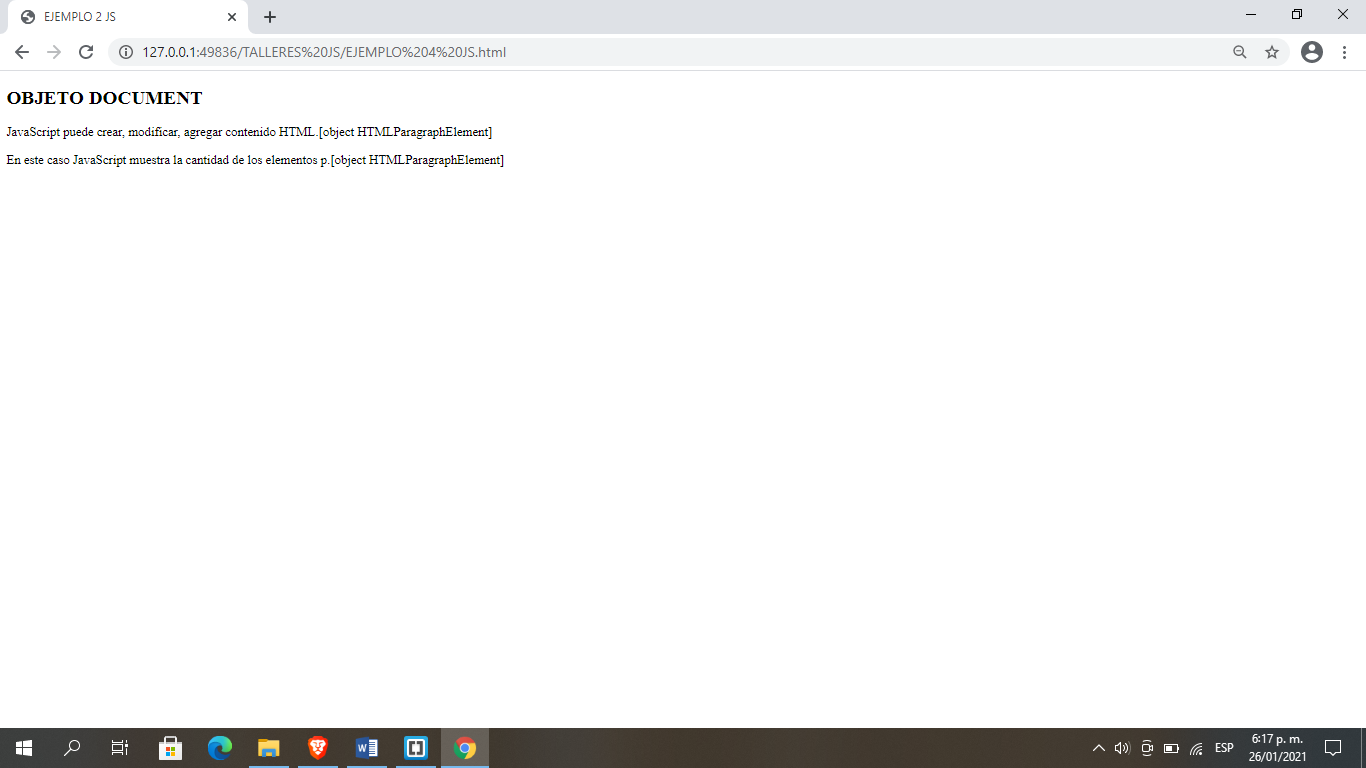
**}**

**</script>**

</body>

</html>

RESULTADO FINAL



## Document.getElementsByName():

El método getElementsByName () se utiliza para buscar elementos del documento HTML con un nombre especificado (atributo name), devuelve un array con todos los elementos, el array se guarda en el objeto **HTMLCollection**.

El objeto **HTMLCollection** contiene un array de elementos o nodos. Podemos acceder a esos nodos mediante los números de índices de su array. El índice comienza en 0.

Como por ejemplo el código que se muestra a continuación:

<!DOCTYPE html>

<html>

<body>

<h1 name="especial">Ejemplo 3</h1>

<p name="especial">En este caso Javascript contará la cantidad de elementos con atributo name=especial </p>

**<script>**

**var nombre= document.getElementsByName("especial").length;**

**document.write(nombre);**

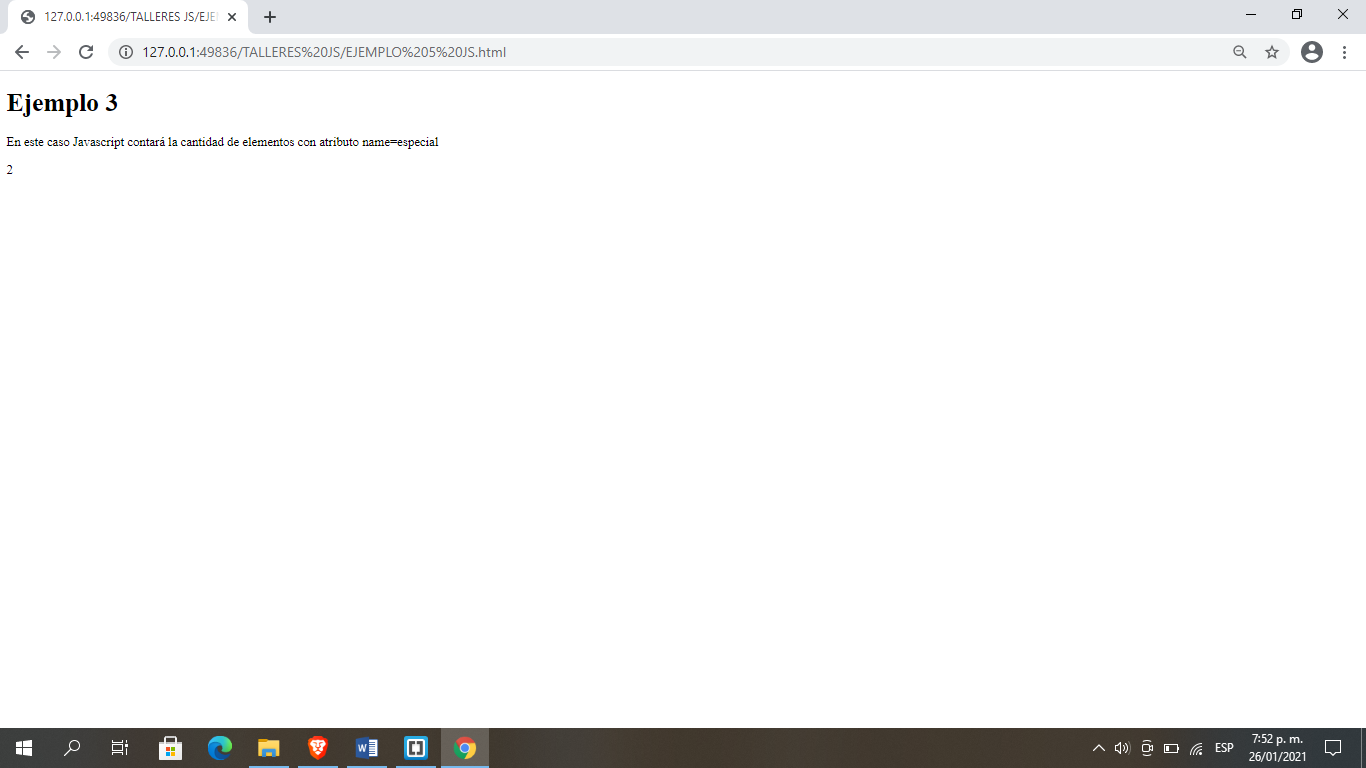
**</script>**

Se crea la variable nombre para buscar y almacenar en la cantidad de elementos con atributo name = ”especial”, dentro de un array, luego muestra en navegador, la cantidad de elementos almacenados en la variable nombre

</body>

</html>

RESULTADO FINAL



* + - 1. **Document.getElementsById():**

Este método podemos considerarlo el que es más utilizado de todos ya que nos permite buscar y acceder de manera directa a los nodos id de la página HTML y de esta manera leer y modificar sus propiedades.

Este método devuelve el nodo cuyo Id coincida con el que se indica en el parámetro de la función, permitiendo hacer únicos cada uno de los nodos de la página HTML.

Se crea dos eventos: un evento onclick para que al dar clic en el botón Globo inflado aparezca el globo normal y otro evento onclick para que al dar clic en botón Globo revienta aparezca reventado por un pincho.

Veamos un ejemplo:

<!DOCTYPE html>

<html>

<body>

<h2>Ejemplo con imagenes - ID</h2>

<button onclick**="document.getElementById('globo').src='img/globo-rojo.jpg'**">Globo inflado-Da clic aquí!!</button>

<img id="globo" src="img/globo-rojo.jpg" style="width:100px">

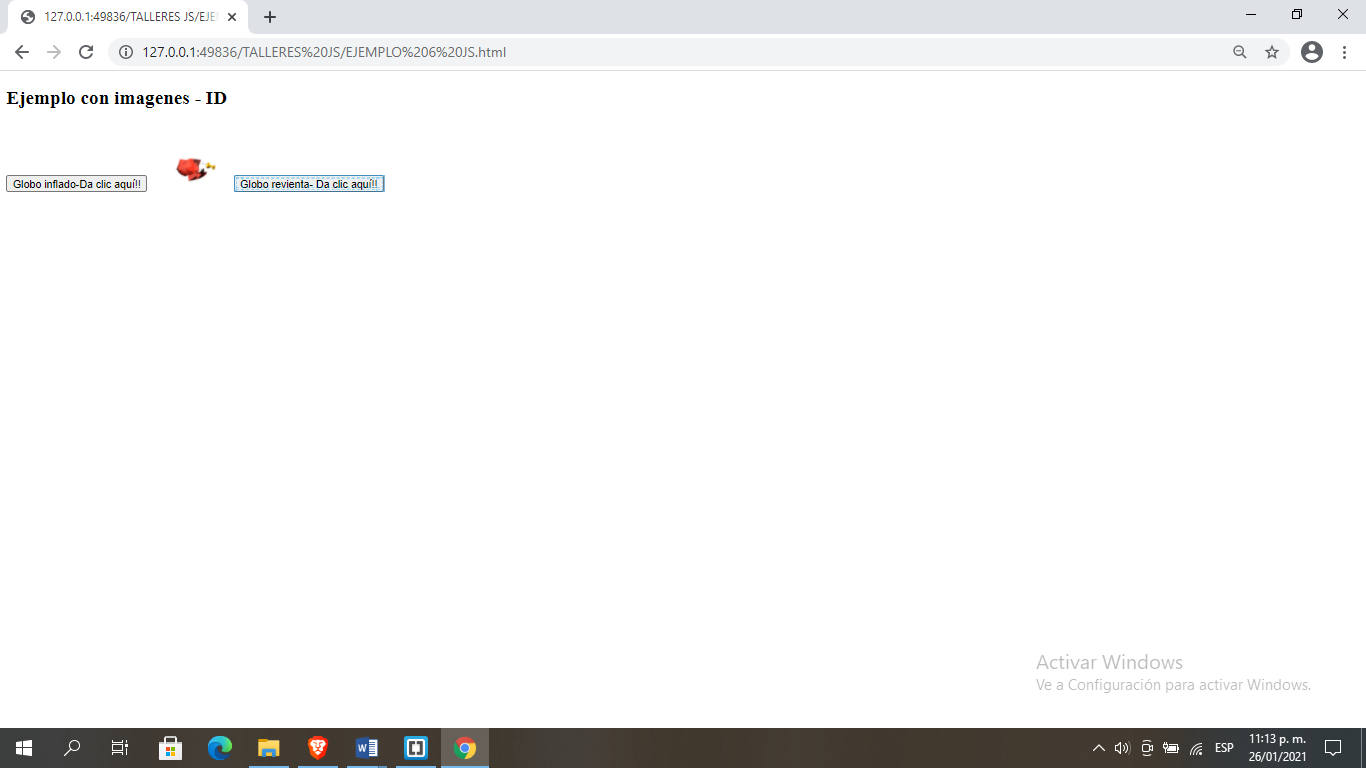
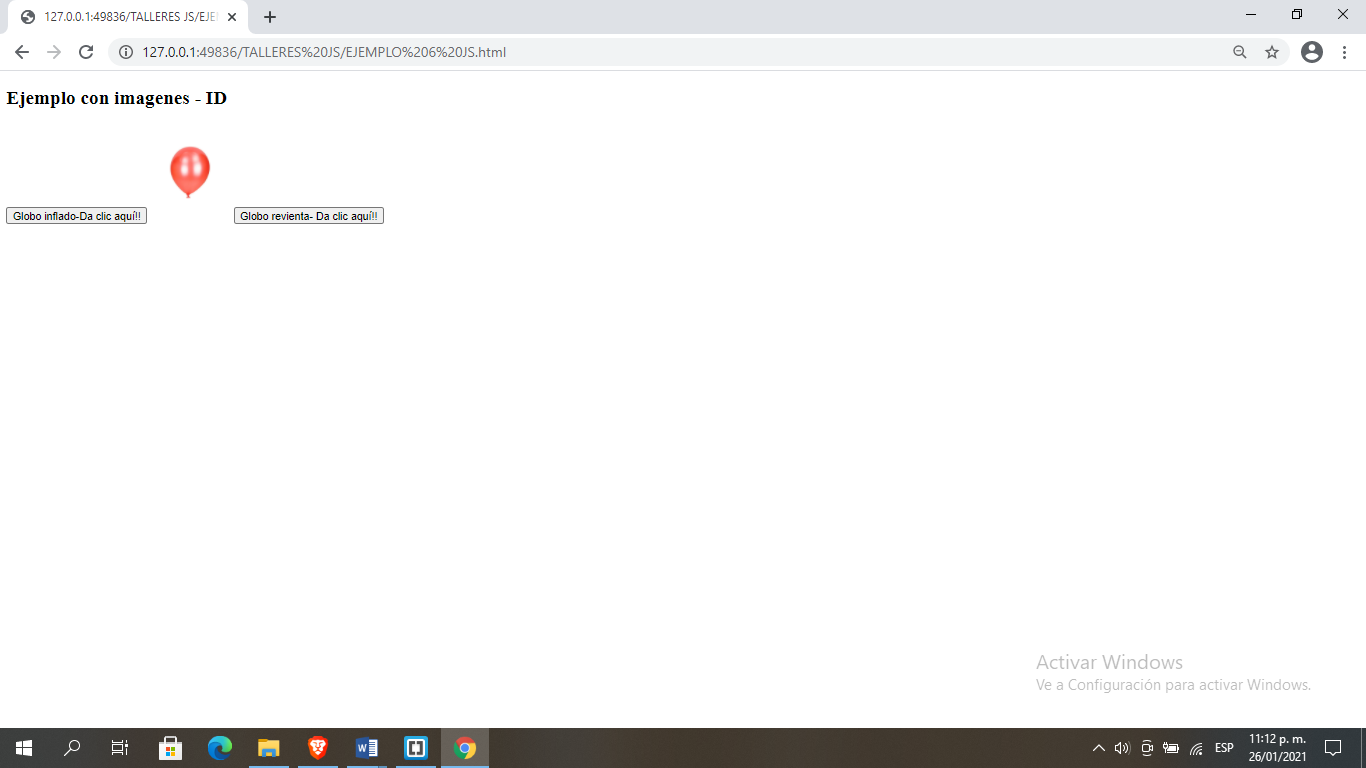
<button onclick="**document.getElementById('globo').src='img/globo-explota.gif'**">Globo revienta- Da clic aquí!!</button>

</body>

</html>

RESULTADO FINAL

Globo normal Globo revienta

**FDEH**

* Utilice el método getElementById para encontrar el elemento que tiene como identificador “HTML” y cambie su texto a "Incap la via del progreso".

<p id="HTML"></p>

<script>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_=”Incap”;

</script>

* Utilice el método getElementsByTagName para encontrar el primer elemento <p> y cambie su texto a "Incap la via del progreso".

<p name="HTML"></p>

<p name="HTML"></p>

<script>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_=”Incap”;

</script>

## Crear Elementos HTML

Para crear elementos HTML requerimos de las siguientes funciones DOM:

1. createElement (Etiqueta HTML): esta función crea un nodo de tipo element, determinado por la etiqueta que se pasa como parámetro en dicha función.
2. createTextNode (contenido): esta función crea un nodo de tipo text almacenando en él el contenido textual de los elementos HTML.
3. parent.appendChild (nodoHijo): determina un nodo como hijo de otro y funciona de la siguiente manera: en primer lugar, se debe añadir el nodo text como hijo del nodo element y luego se añade el nodo element como hijo del nodo que representa el elemento de la página HTML.

**EJEMPLO**

En este ejemplo aplicamos la creación de elementos y nodos de texto, se declaran las constantes para asignar los <id> de elementos <h1>y<ul>, luego se crea el elemento <li> mediante **document.createElement**, luego se agrega al elemento <li> el nodo de texto Enero, luego se agrega al elemento padre <ul> el mes de Enero. Por último, se imprime en pantalla un título <h1> y se llama a la consola para mostrar var lista

<html>

<head>

<meta charset="UTF-8">

</head>

<body>

<h1 id="**cabecera**"></h1>

<ul id="**mes**"></ul>

//

<script type="text/javascript">

**const cabecera=document.getElementById("cabecera");**

**const mes=document.getElementById("mes");**

**const lista=document.createElement("li");**

**lista.textContent='Enero';**

**mes.appendChild(lista);**

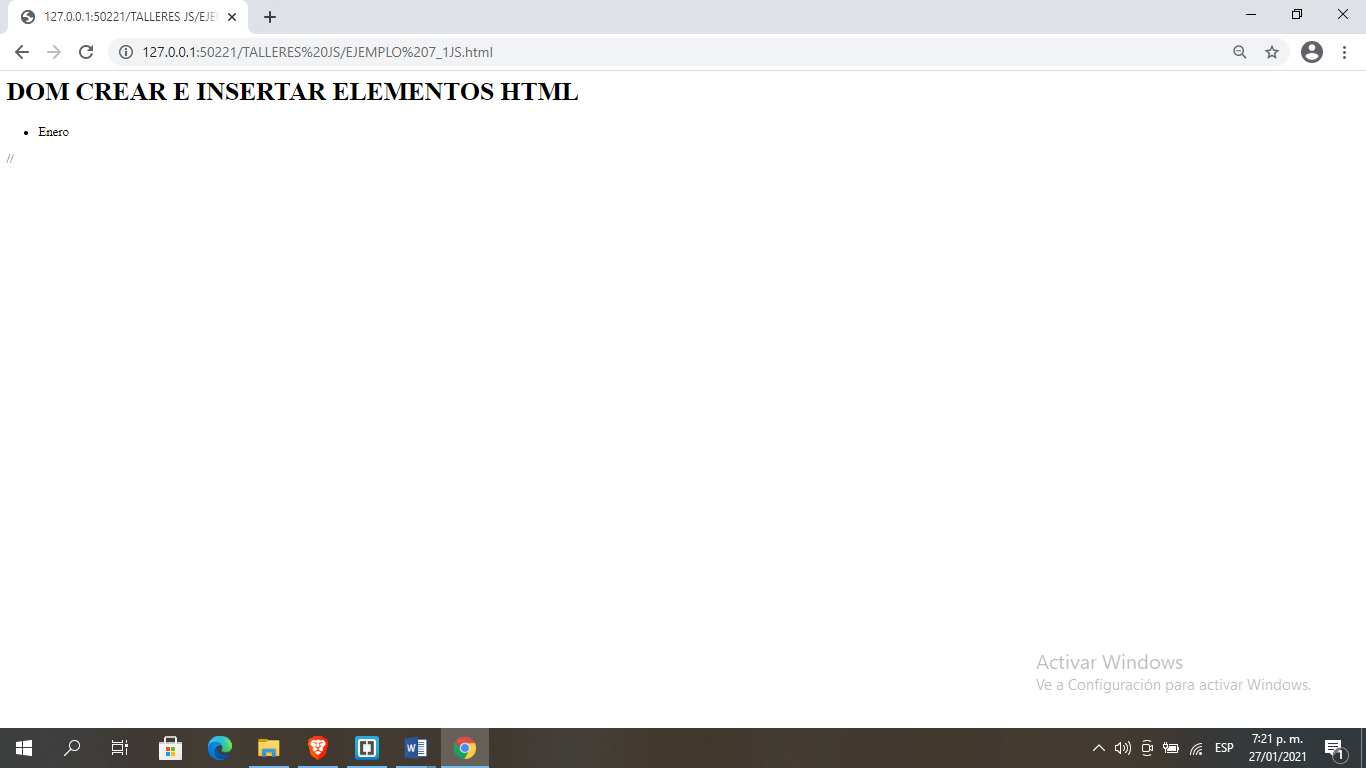
**cabecera.innerHTML='DOM CREAR E INSERTAR ELEMENTOS HTML';**

**console.log(lista);**

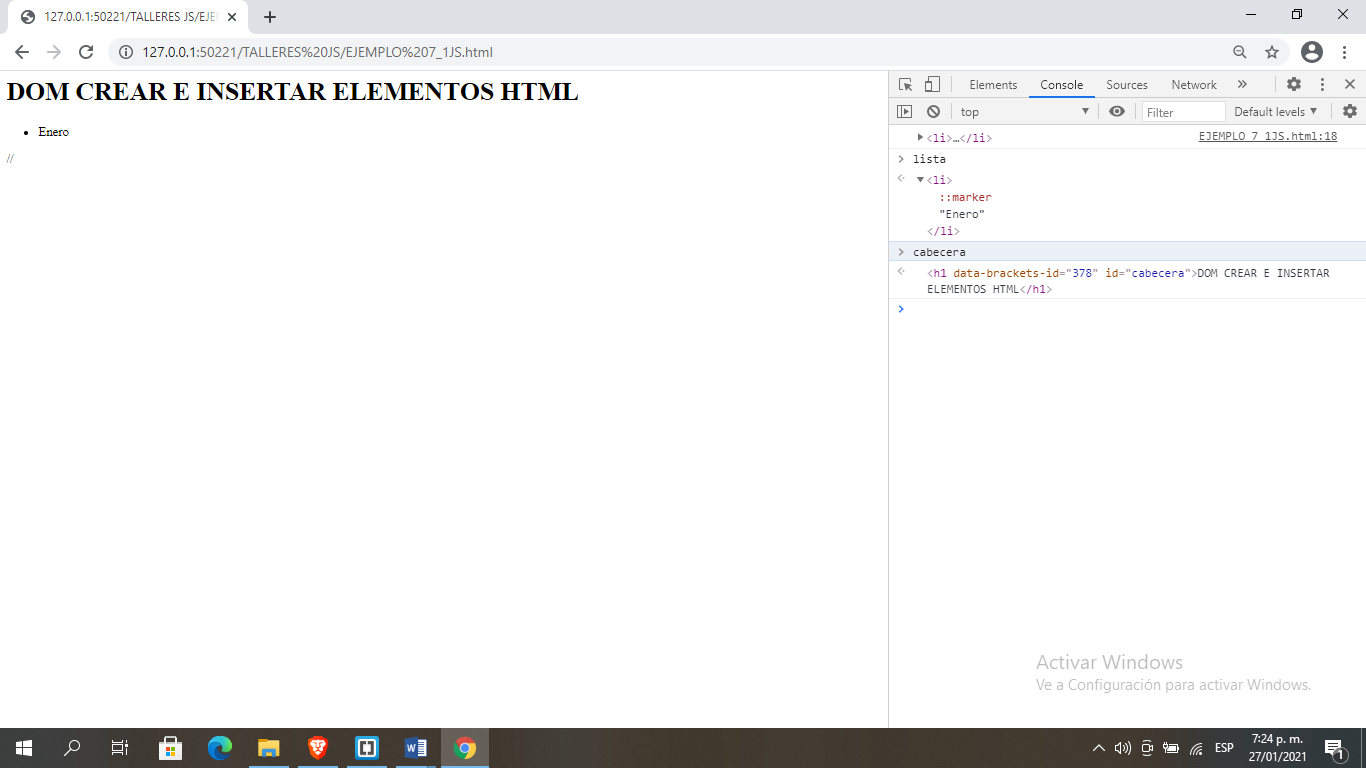
</script>

</body>

</html>



**Ventana consola**



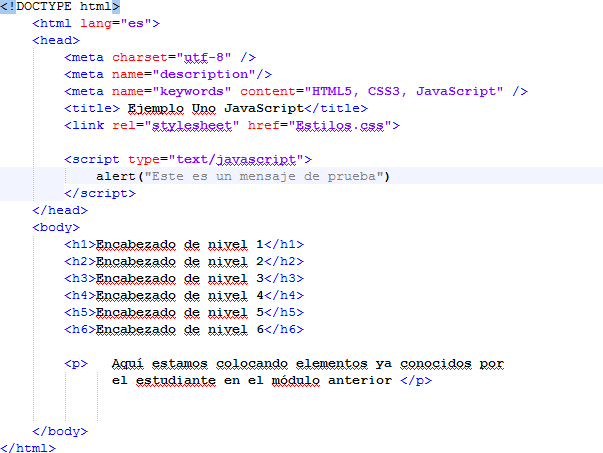
Miremos el caso de si se quiere añadir un encabezado simple dentro de una página HTML, el script que debería utilizarse es el siguiente:

## Como incluir JavaScript en documentos HTML

## 

Para realizar esta operación el código se debe encerrar entre las etiquetas <script> y </script> y se puede agregar en cualquier parte del documento, aunque se recomienda incluirlo dentro de la etiqueta de encabezado de la página <head>; más adelante se dan algunos consejos adicionales al respecto según sea el caso.

Miremos un pequeño ejemplo:



En el ejemplo anterior podemos observar encerrado entre un rectángulo de color rojo el script embebido en el documento HTML. Esta forma de trabajar con JavaScript tiene la desventaja de si se presenta la necesidad de realizar cambios en el código, sería necesario realizar la modificación en todas las páginas que dependan del mismo bloque de código. Pero tiene como ventaja que lo podemos utilizar cuando se requiere generar procesos específicos en una página en particular y cuando manejamos pequeños bloques de código.

## JavaScript definido en un documento externo

El código de JavaScript se puede incluir en un archivo aparte, teniendo en cuenta que se debe guardar con la extensión js; a este archivo las paginas HTML se enlazan mediante las etiquetas

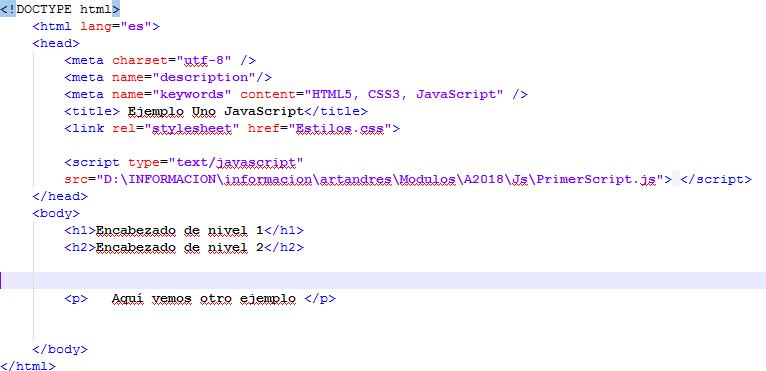
<script> y </script> y podemos crear todos los archivos JavaScript que se requieran si es el caso.

Veamos un ejemplo:

El código JavaScript será:



Este script se debe guardar con la extensión js. Y la forma de enlazar la página HTML con el archivo anterior es:



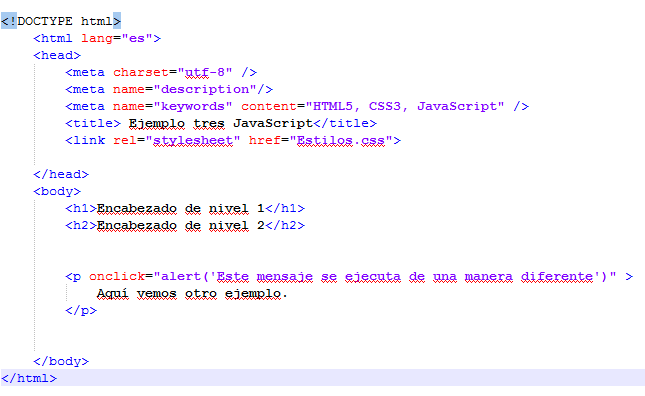
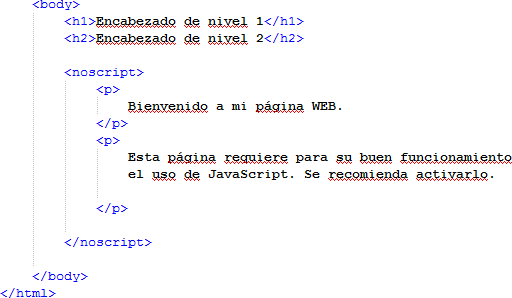
Podemos observar que adicionalmente al atributo type se debe utilizar de igual manera el atributo src, el cual indica la ruta en donde se encuentra el archivo .js o JavaScript creado y con el cual se enlaza la página HTML; cabe anotar que cada etiqueta <script> enlaza un único archivo de JavaScript, pero en una página Web se pueden utilizar un número indeterminado de estas etiquetas si es el caso.

La mayor ventaja de utilizar este método es respecto al mantenimiento ya que minimiza o disminuye el código que se debe utilizar en la página HTML y el código JavaScript lo podemos reutilizar en otras páginas dependiendo de nuestras necesidades.

## Incluir JavaScript en los elementos HTML

Esta forma de utilizar JavaScript consiste en incluir partes de código JavaScript dentro del código HTML. La dificultad más grande que representa este método es que por decirlo de alguna manera viciamos el código HTML y se complica el mantenimiento del código escrito en JavaScript; de tal manera que este método no se recomienda y en nuestro caso no lo utilizaremos.

Veamos un ejemplo:



Para terminar si por alguna circunstancia el usuario de una página WEB deshabilita el uso de JavaScript es razonable que podamos advertir de alguna manera que la pagina requiere de su uso; esto se hace utilizando la etiqueta <noscript> de la siguiente manera:

**FDEH**

* El estudiante creara un script que permita mostrar sus datos personales y una bienvenida a sus compañeros de clase.
* El estudiante creara un script utilizando arrays que permita almacenar los meses del año y luego muestre el mes de febrero.
* El estudiante creara un script que permita almacenar en un arreglo la siguiente información: [5, true, false, 3, “Hola Amigos”, Hola INCAP”] y con ella realizar las siguientes operaciones:
  + Determinar cuál de los elementos de texto es mayor, solo utilizando lo aprendido hasta ahora.
  + Utilizando solo los valores booleanos determinar las operaciones necesarias para que el resultado sea true.
  + Realizar las cuatro operaciones básicas matemáticas con los valores numéricos.

# UNIDAD 2

# METODOS, EVENTOS Y PROPIEDADES DE JAVASCRIPT

Utilizar de manera técnica dentro de la programación de una interfaz de usuario, los métodos, Condicionales, Bucles y funciones en JavaScript y generar de esta forma la interfaz gráfica de usuario adecuada.

**FDH**

## Propiedad innerHTML

Esta propiedad se utiliza para almacenar el contenido de un elemento HTML, por lo general se utiliza un document.getElementById para posteriormente mediante la instrucción alert o console.log mostrar el contenido almacenado en el navegador web.

Ejemplo 1:

Se crea la variable **conten** para asignarle el contenido del elemento <p> con id=”**texto**” para después mostrarlo por medio de un cuadro de dialogo.

<html>

<head>

<meta charset="UTF-8">

</head>

<body>

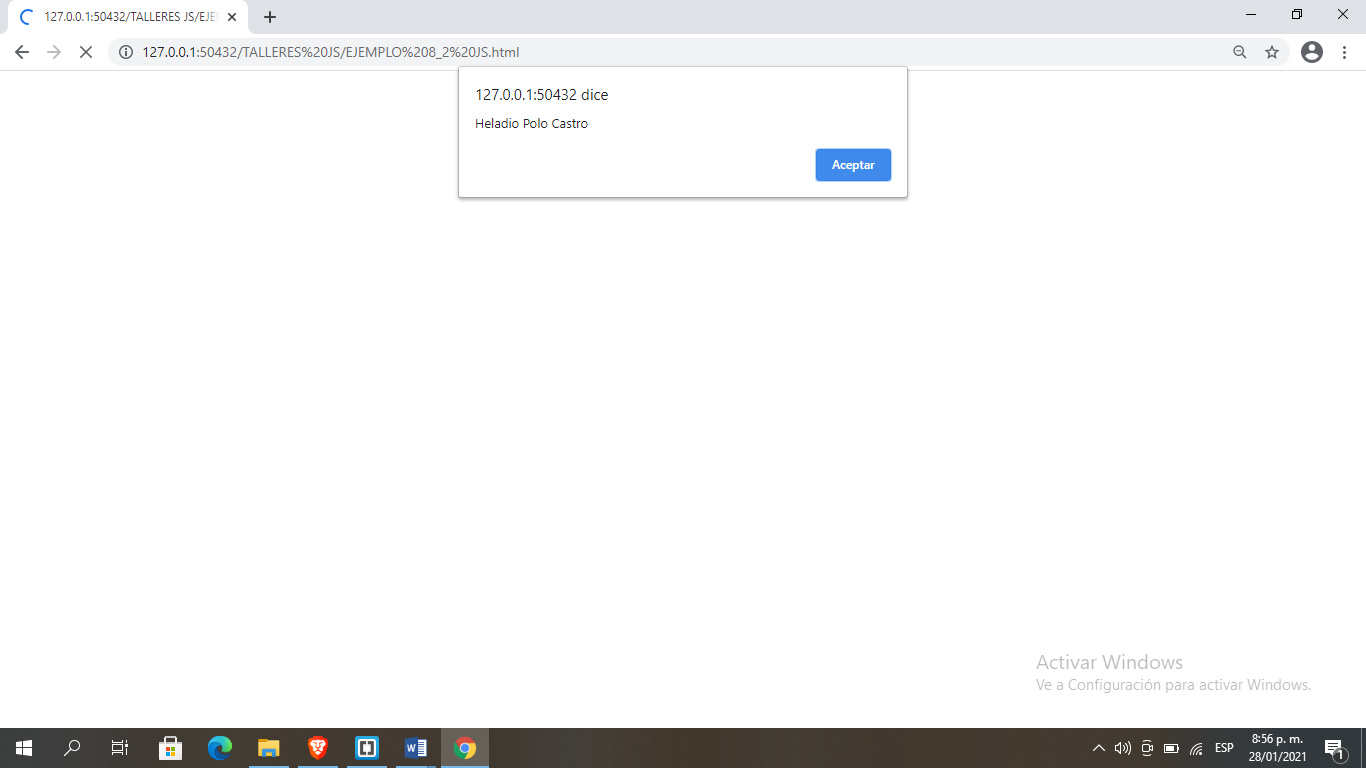
<p **id="texto">**Heladio Polo Castro</p>

**//**

<script type="text/javascript">

var conten= **document.getElementById("texto").innerHTML;**

alert(conten);

</script>

**</body>**

**</html>**

Se asigna la cadena de texto indicada al contenido del elemento <p> con id=”**texto**” para después mostrarlo por la salida del navegador web.

***Ejemplo 2***

<html>

<head>

<meta charset="UTF-8">

</head>

<body>

<p id="**texto**"></p>

//

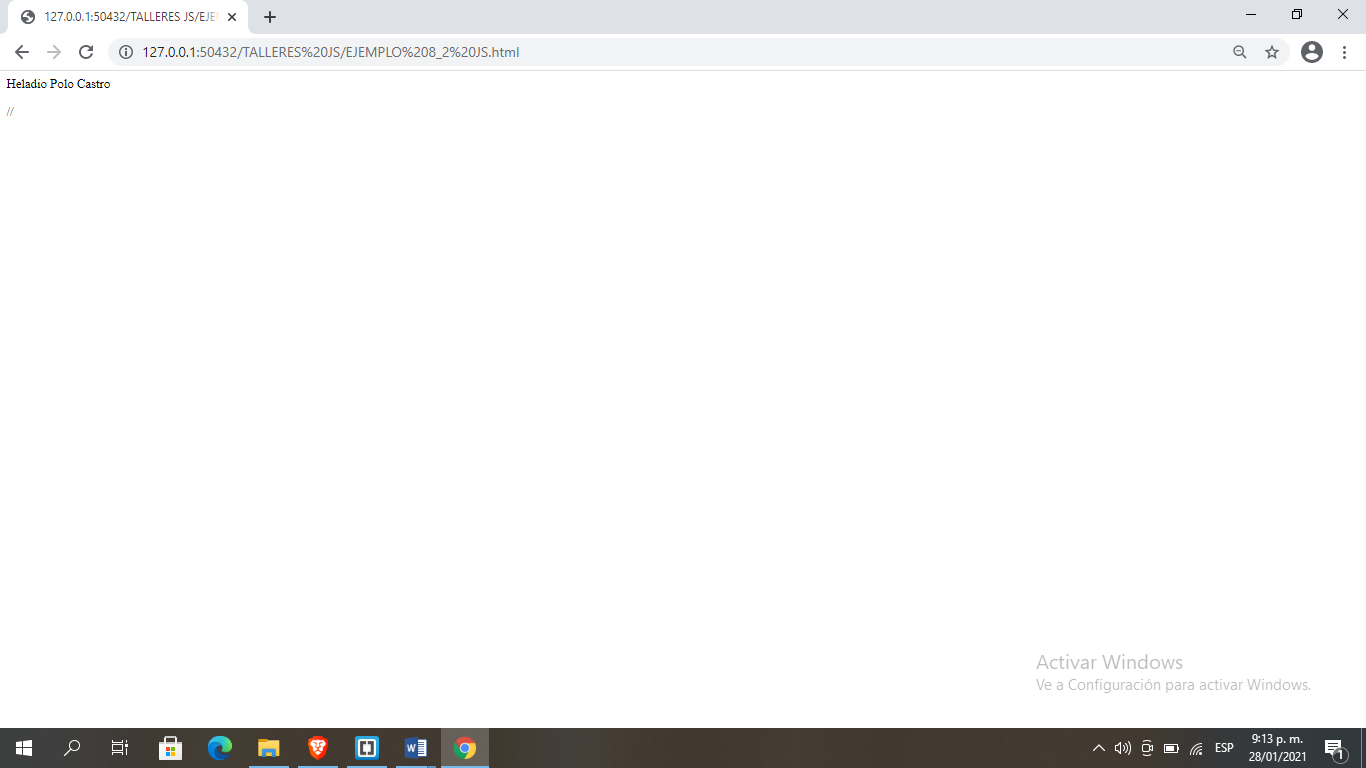
**<script type="text/javascript">**

**document.getElementById("texto").innerHTML="Heladio Polo Castro";**

**</script>**

</body>

</html>



* 1. **Métodos String de Javascript**

Estos métodos se utilizan para gestionar cadenas de texto.

* + 1. **Método length**

Se utiliza para generar el largo de una cadena de texto.

<script type="text/javascript">

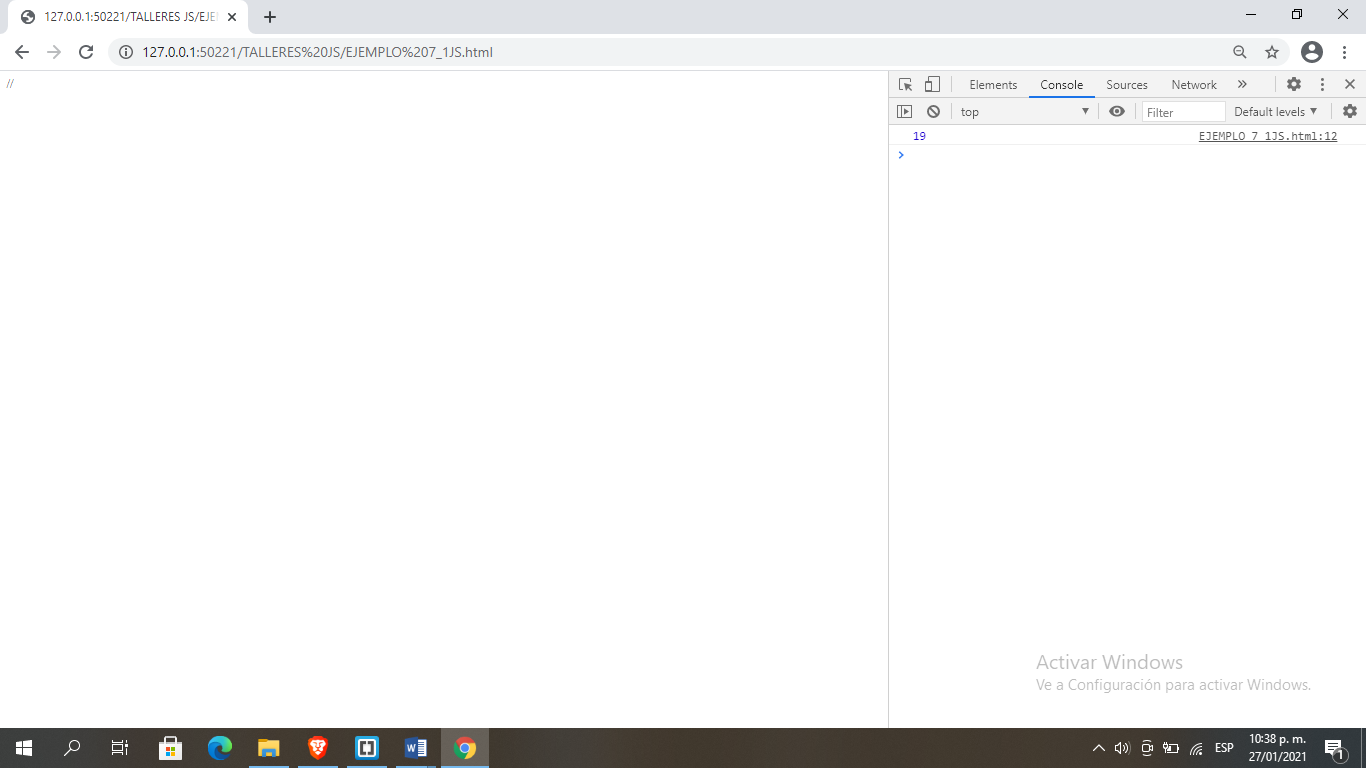
Vamos a generar la cantidad de caracteres que tiene la cadena de texto nombre y apellidos, muestra 19

var b="Heladio Polo Castro";

var largo=b.**length**;

console.log(largo);

</script>



* + 1. **Método concat()**

Se utiliza para unir dos o más strings, se puede utilizar el operador + junto con las comillas simples ’ ‘ para unir dos valores.

**Ejemplo**

document. write(**nombre + ' tu materia es aprobada con un… ' + nota**);

document.write(‘nombre estudiante’+’ ‘+nombre);

* + 1. **Método Indexof()**

Devuelve el índice de la posición de la primera letra de la cadena de texto a buscar, aclarando que la búsqueda comienza de izquierda a derecha a partir del índice 0, luego 1,2,3,….n.

Vamos a generar la posición donde encuentra el apellido Polo en la cadena de texto, muestra 8

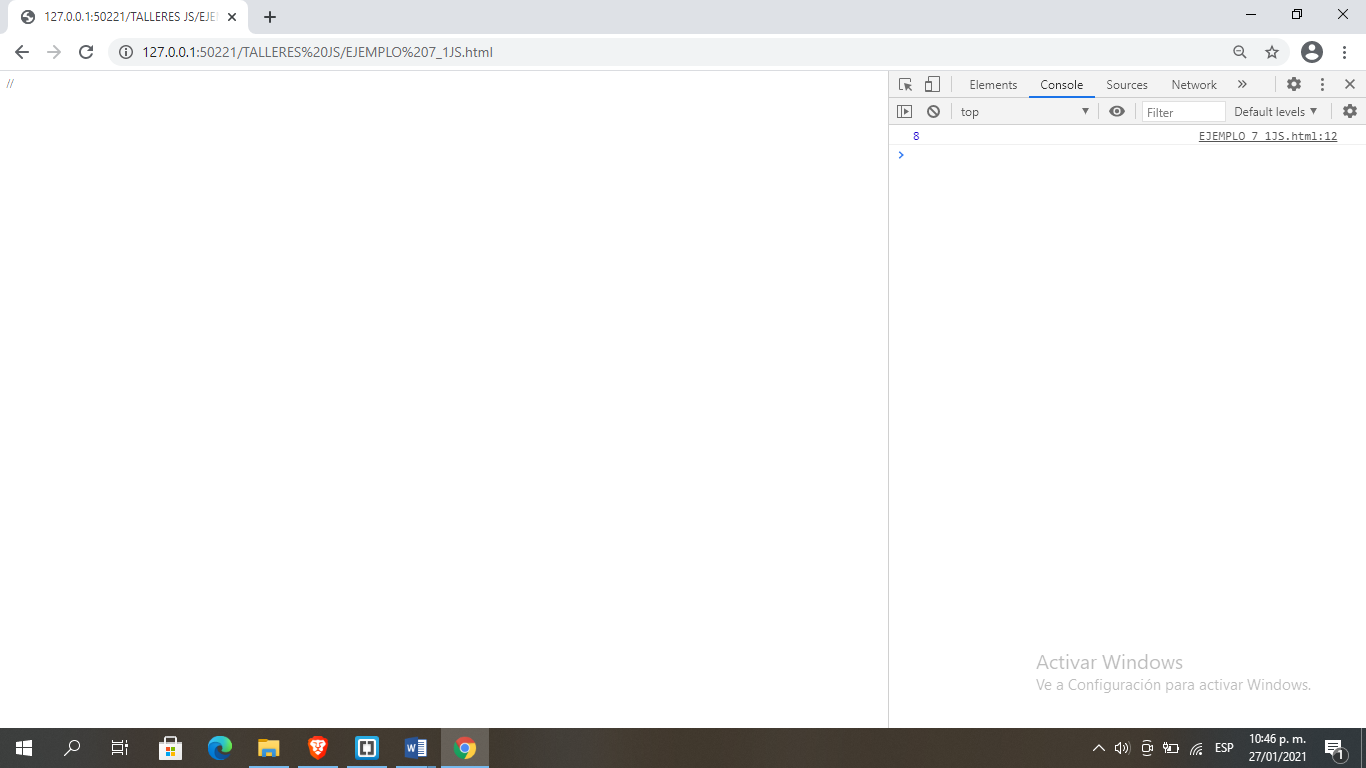
<script type="text/javascript">

var b="Heladio Polo Castro";

var largo=b.**indexOf**("Polo");

console.log(largo);

</script>



* + 1. **Método Substring()**

Permite hallar de acuerdo a la posición inicial y la posición final la cadena de texto resultante.

Vamos a generar la cadena de texto de acuerdo a su ubicación dando la posición Inicial y la posición final, muestra Heladio

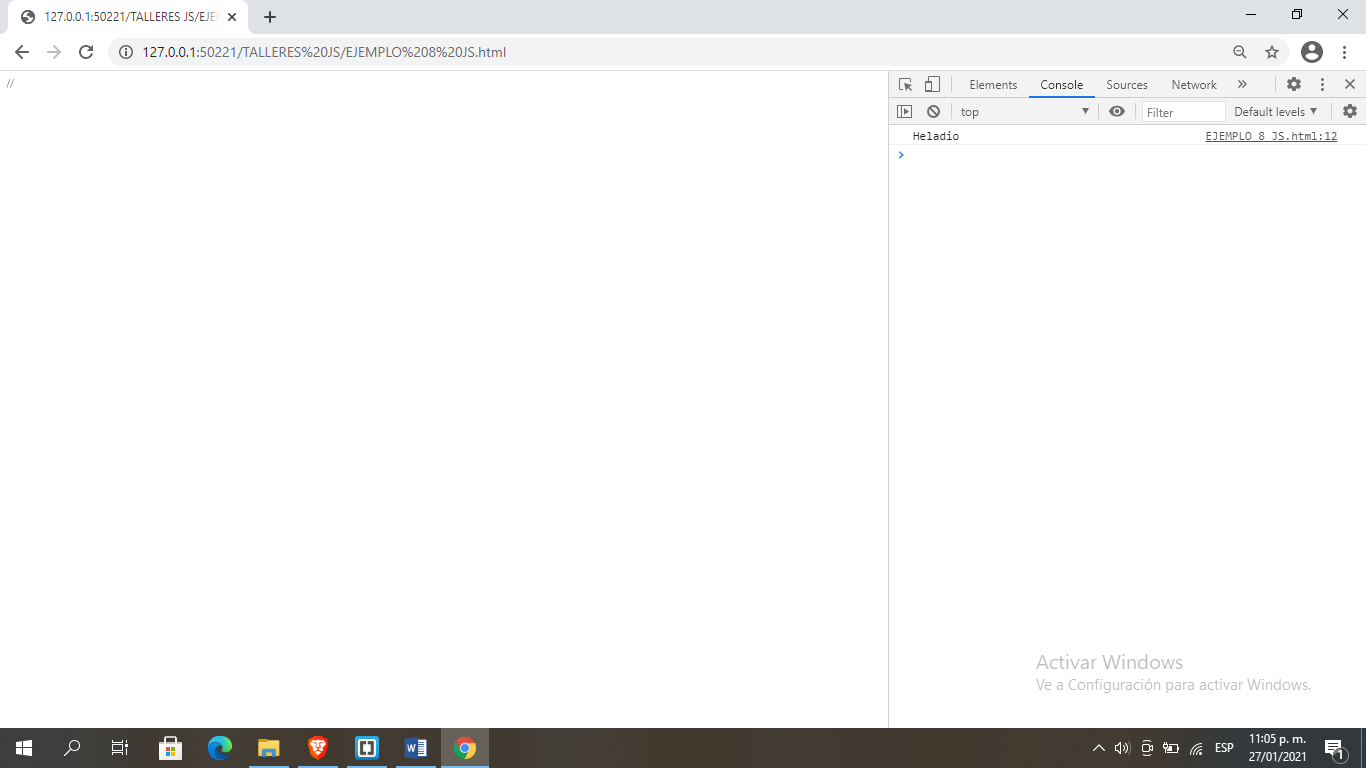
<script type="text/javascript">

var b="Heladio Polo Castro";

var largo=b.**substring**(0,7);

console.log(largo);

</script>



* + 1. **Método Replace()**

Permite sustituir un nuevo texto dentro de una cadena de texto previa localización.

Vamos a reemplazar el texto Heladio por el texto Mello, muestra Mello Polo Castro

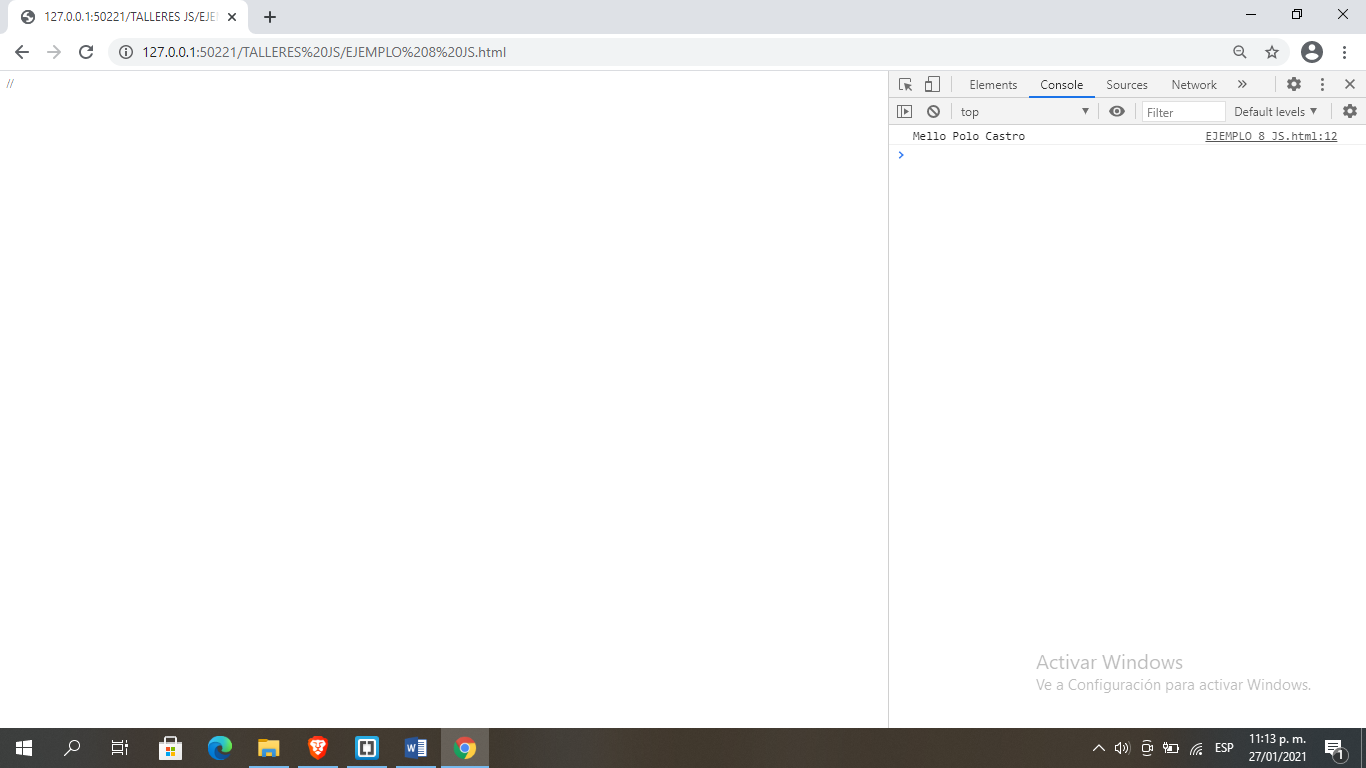
<script type="text/javascript">

var b="Heladio Polo Castro";

var largo=b.**replace**("Heladio","Mello");

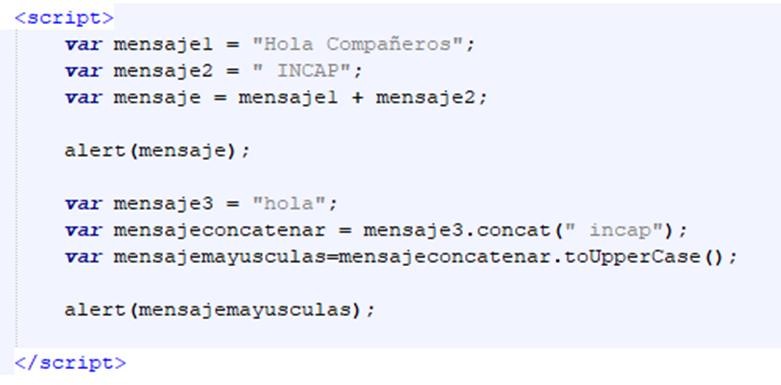
console.log(largo);

</script>



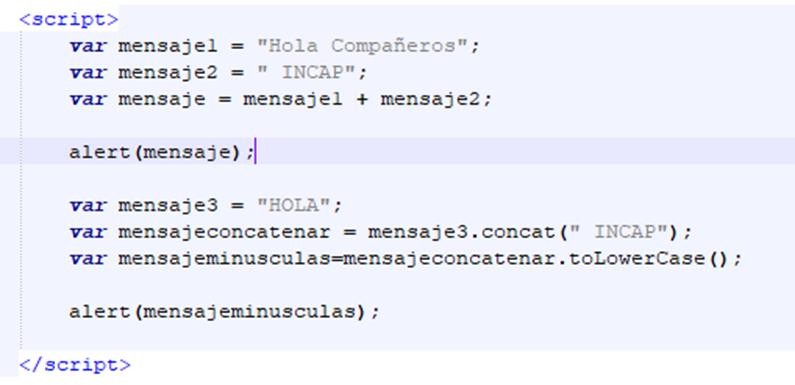
* + 1. **Función toUpperCase**

Transforma los caracteres en mayúscula. Ejemplo:



* + 1. **Función toLowerCase**

Transforma o convierte los caracteres en minúscula si se encuentran en mayúscula.



Ejemplo:

A pesar de que existen muchas más funciones para manejo de cadena de caracteres tratamos solo las anteriores ya que son las más utilizadas; las demás quedaran a discreción del formador para estudiarlas y aplicarlas.

* 1. **Métodos Numéricos de Javascript**

Estos métodos se utilizan para gestionar valores numéricos.

* + 1. **Método toString()**

Se utiliza para convertir un numero en una cadena de texto.

Ejemplo:

<script type="text/javascript">

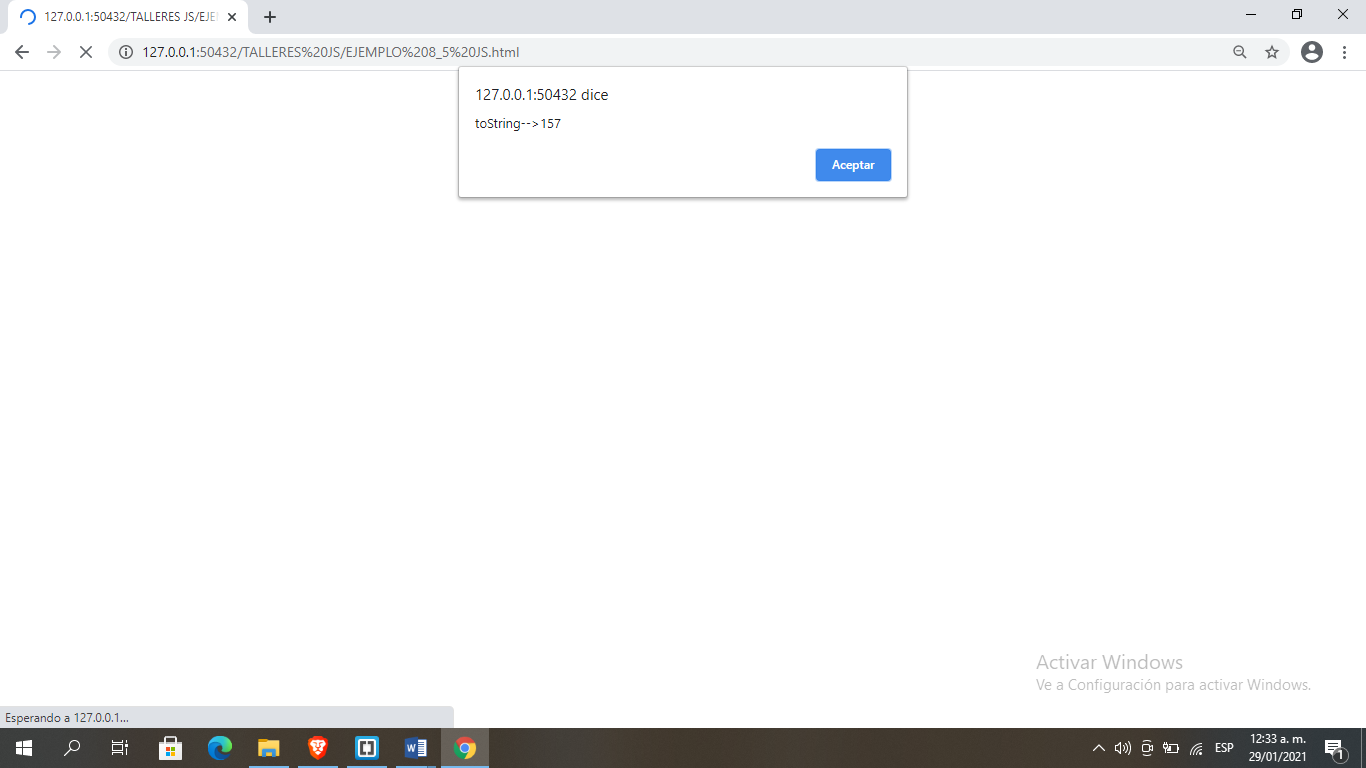
En este ejemplo el valor numérico 157 se convierte en un texto

var numx = 157;

var num1= numx.**toString**();

alert("toString-->"+num1);

</script>



* + 1. **Método toFixed()**

Se utiliza para generar un numero con decimales dependiendo de los números decimales solicitados en expresión si es 0 se aproxima al número más alto siguiente.

En este ejemplo el valor numérico 15.757 utilizando el método toFixed y el parámetro 0 lo aproxima al número siguiente más alto =16

Ejemplo:

<script type="text/javascript">

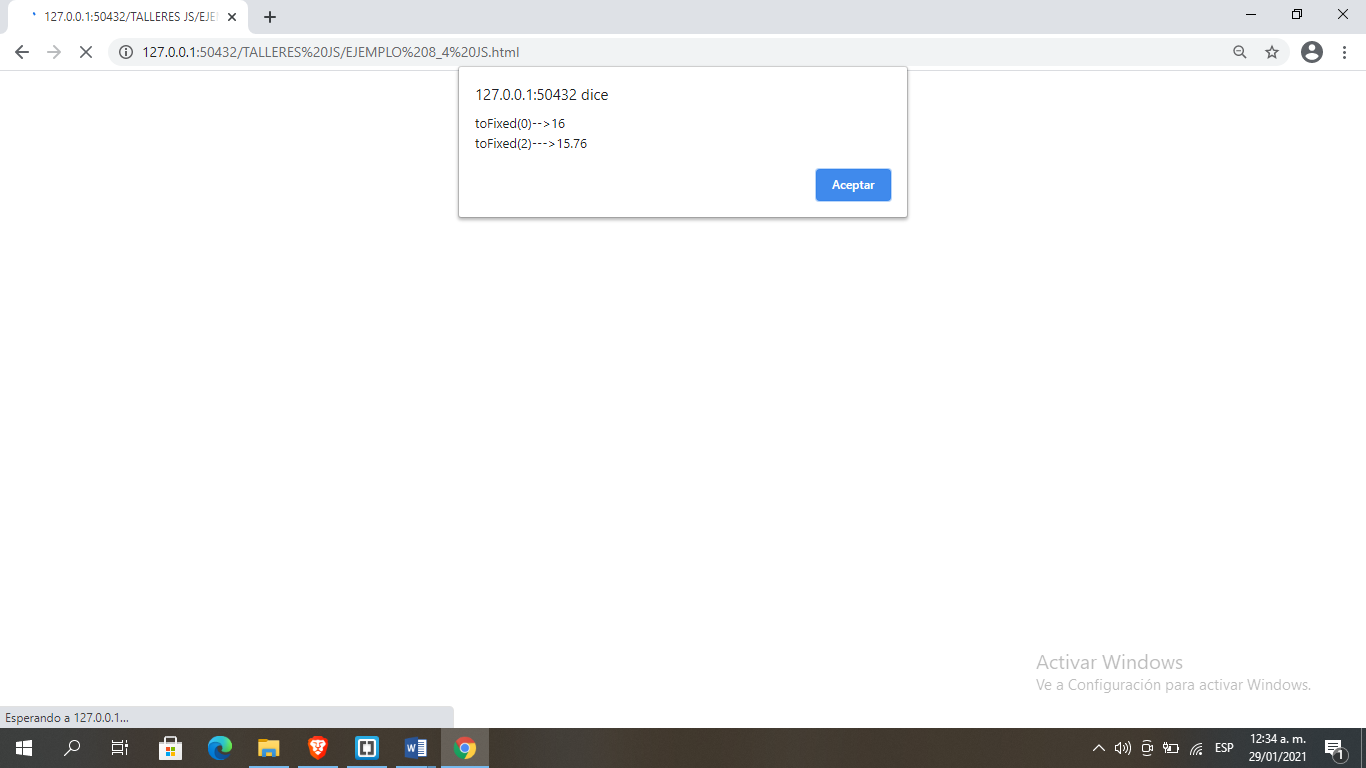
var numx = 15.757;

var num1= numx.**toFixed(0);**

var num2= numx.**toFixed(2);**

alert("toFixed(0)-->"+num1+" "+"\ntoFixed(2)--->"+num2);

</script>



* + 1. **Método parseInt**

Se utiliza para convertir a numero una cadena de texto.

En este ejemplo el método parseInt convierte la cadena de texto 15 años a un número verdadero.

<h1>Metodo parseInt</h1>

<p id="texto"></p>

//

<script type="text/javascript">

var numx = "15 años";

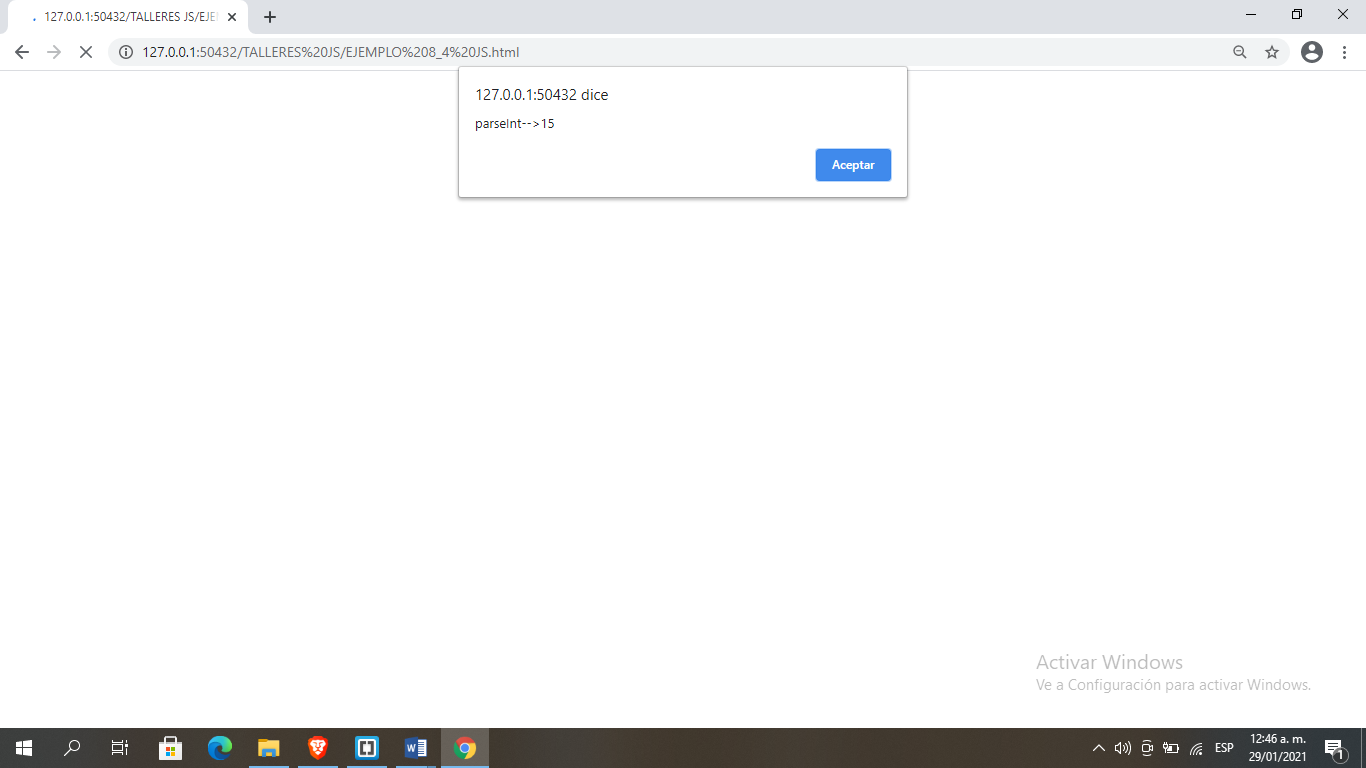
var num1= **parseInt**(numx);

//

document.getElementById("texto").innerHTML= num1;

alert("parseInt-->"+num1);

</script>



* + 1. **Método isNaN**

Devuelve falso si un numero correcto o verdadero si no es un número o es un expresión matemática incorrecta.

Ejemplo:

En este ejemplo el método isNaN verifica si el dato 15 años no es un número y su resultado es verdadero. (es una cadena de texto)

<p id="texto"></p>

//

<script type="text/javascript">

var numx = "15 años";

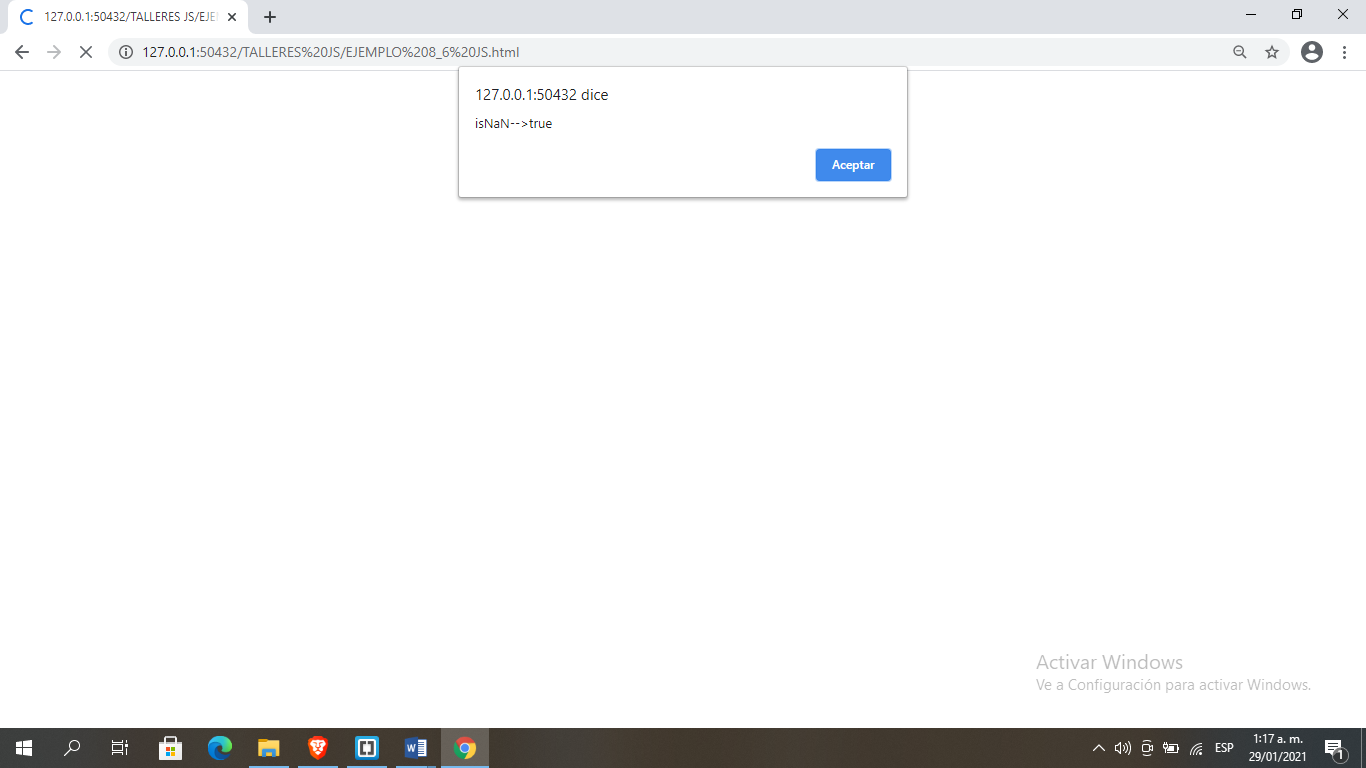
var num1= **isNaN**(numx);

//

document.getElementById("texto").innerHTML= num1;

alert("isNaN-->"+num1);

</script>



* 1. **Métodos Arrays de Javascript**
     1. **Método toString()**

Este se utiliza para convertir los valores integrantes de un arreglo vector en un array de strings.

Ejemplo

<h1 id="cabecera">METODO ARRAYS EN JAVASCRIPT</h1>

<p id="texto"></p>

<script type="text/javascript">

var mes\_list =['Enero','Febrero','Marzo','Abril','Mayo','Junio','Julio','Agosto','Septiembre','Octubre','Noviembre','Diciembre'];

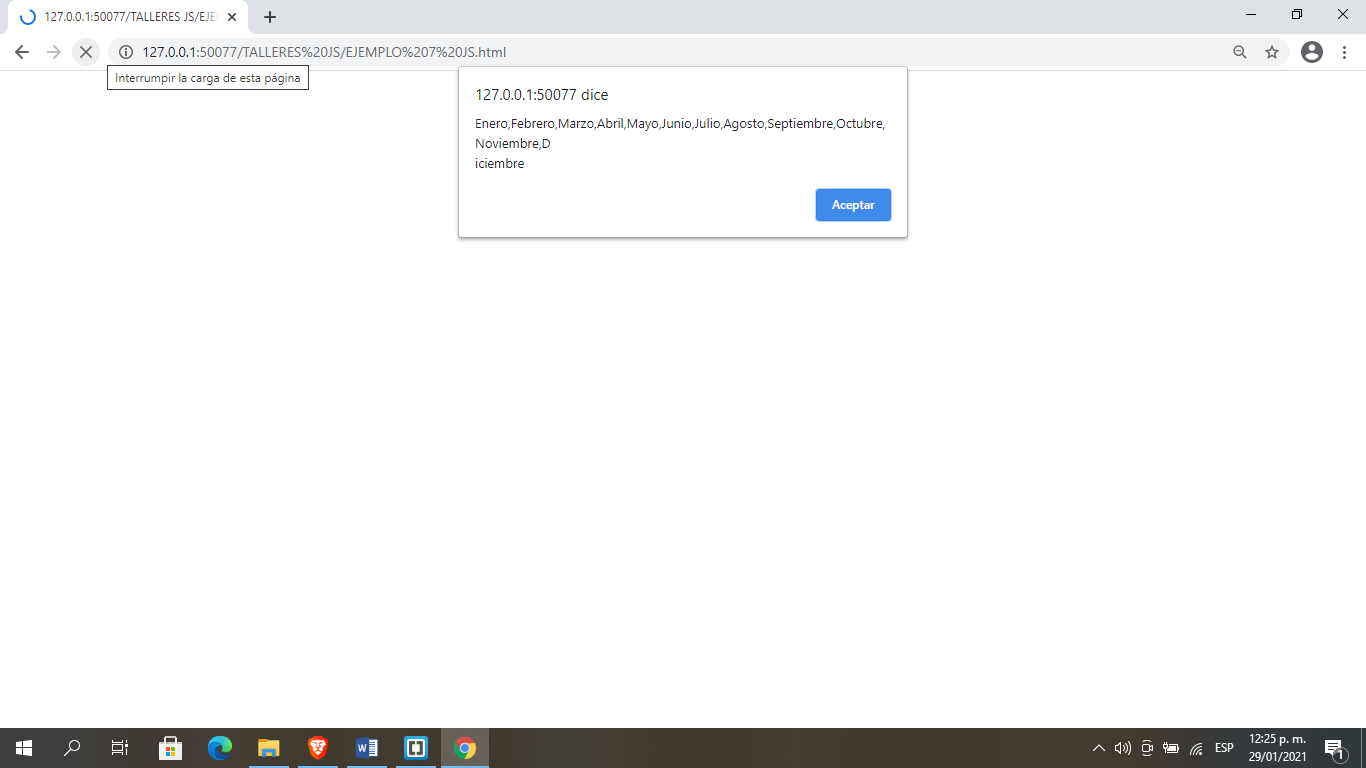
document.getElementById("texto").innerHTML=mes\_list.**toString()**;

var lista = mes\_list.**toString()**;

alert(lista);

</script>

En este ejemplo el método toString( ) se utiliza para convertir los elementos de un Arrays en un arreglo de textos



* + 1. **Método POP()**

Método que se utiliza eliminar el último componente de un arreglo tipo vector.

Ejemplo

<h1 id="cabecera">METODO POP EN JAVASCRIPT</h1>

<p id="texto"></p>

<script type="text/javascript">

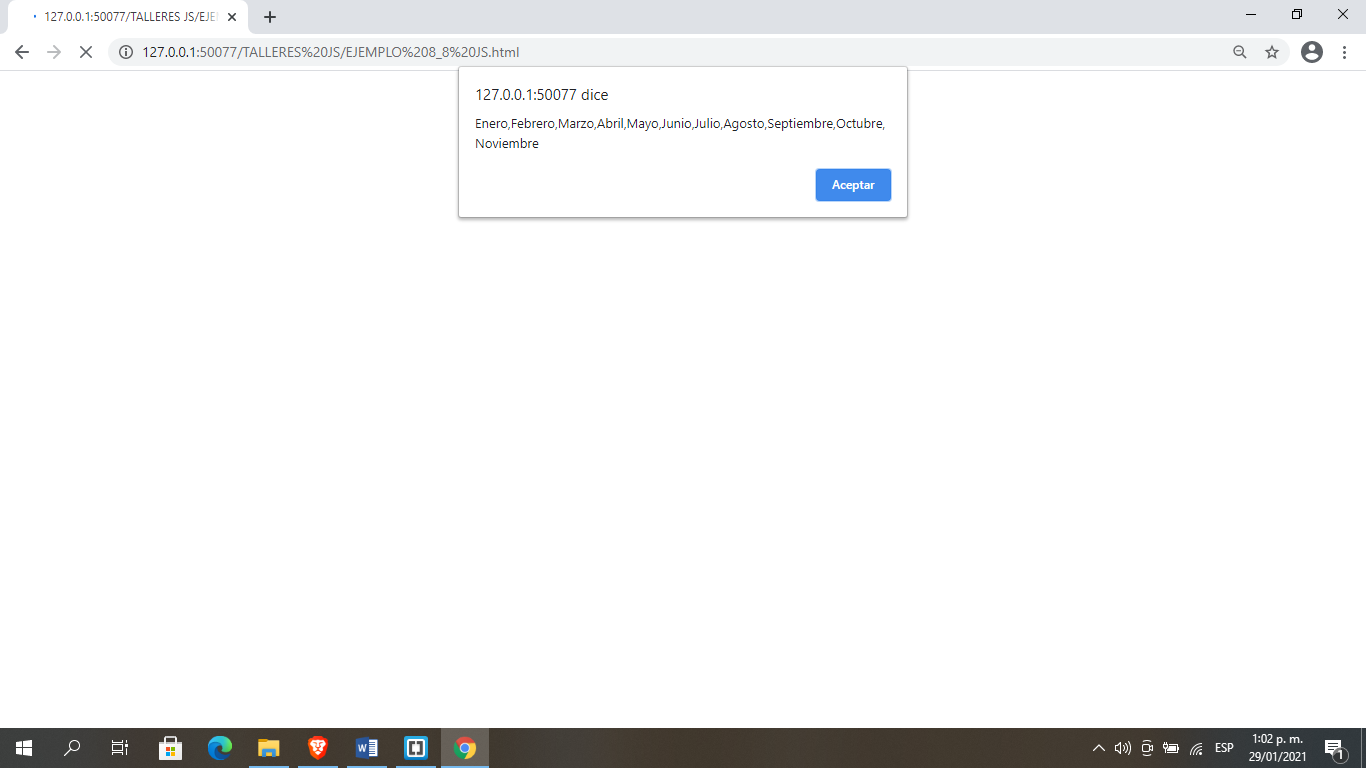
var mes\_list =['Enero','Febrero','Marzo','Abril','Mayo','Junio','Julio','Agosto','Septiembre','Octubre','Noviembre','Diciembre'];

document.getElementById("texto").innerHTML=mes\_list**.pop();**

var lista = mes\_list;

alert(lista);

</script>



En este ejemplo el método pop( ) se utiliza para eliminar el último elemento (Diciembre) de un array.

* + 1. **Método PUSH()**

Método que se utiliza agregar un componente a un arreglo tipo vector (al final).

**Ejemplo**

<h1 id="cabecera">METODO PUSH EN JAVASCRIPT</h1>

<p id="texto"></p>

<script type="text/javascript">

var mes\_list =['Enero','Febrero','Marzo','Abril','Mayo','Junio','Julio','Agosto','Septiembre','Octubre','Noviembre'];

document.getElementById("texto").innerHTML=mes\_list.**push('Diciembre');**

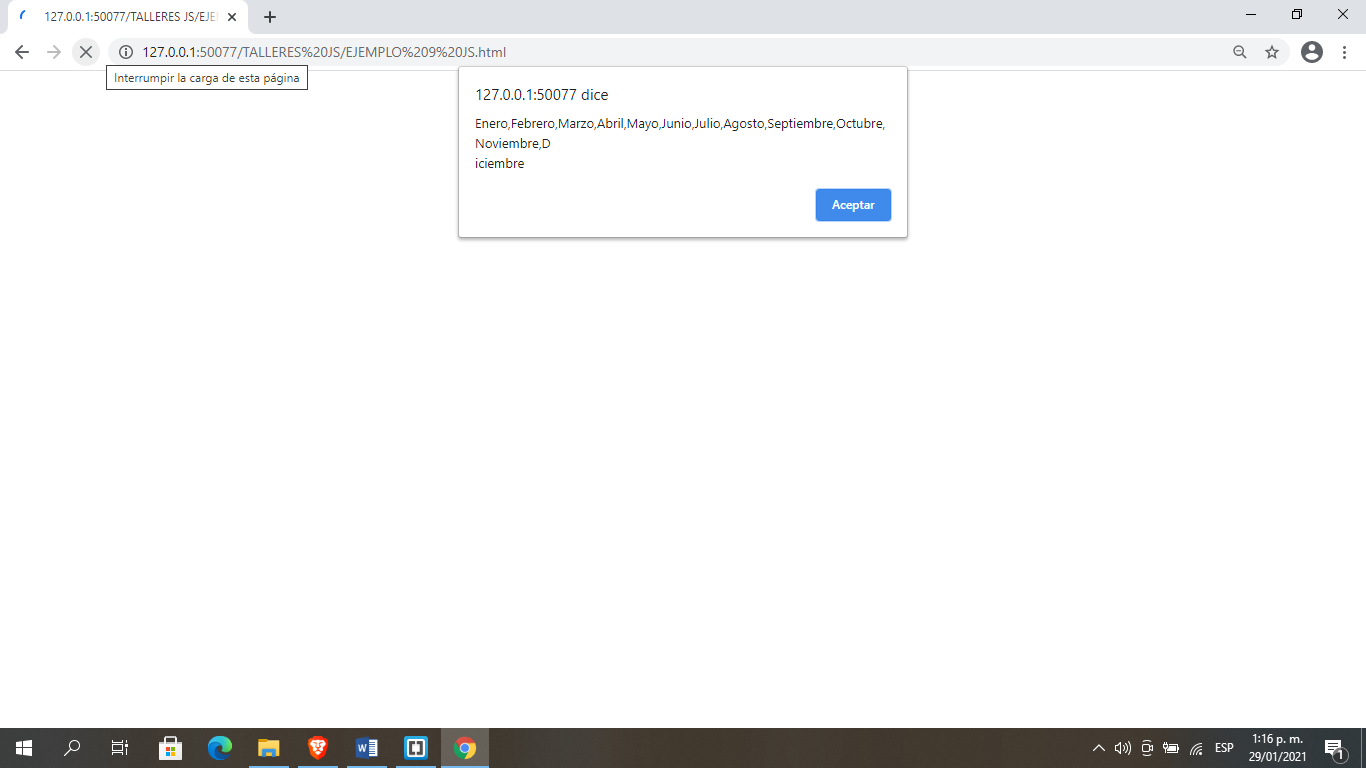
var lista = mes\_list;

//

alert(lista);

</script>

En este ejemplo el método push( ) se utiliza para agregar el elemento (Diciembre) al final de un array.



* + 1. **Método SHIFT()**

Método que se utiliza eliminar un componente a un arreglo tipo vector (al principio), a su vez nos muestra el componente del array que se suprimió.

Ejemplo

<h1 id="cabecera">METODO SHIFT EN JAVASCRIPT</h1>

<p id="texto"></p>

<script type="text/javascript">

var mes\_list =['Enero','Febrero','Marzo','Abril','Mayo','Junio','Julio','Agosto','Septiembre','Octubre','Noviembre','Diciembre'];

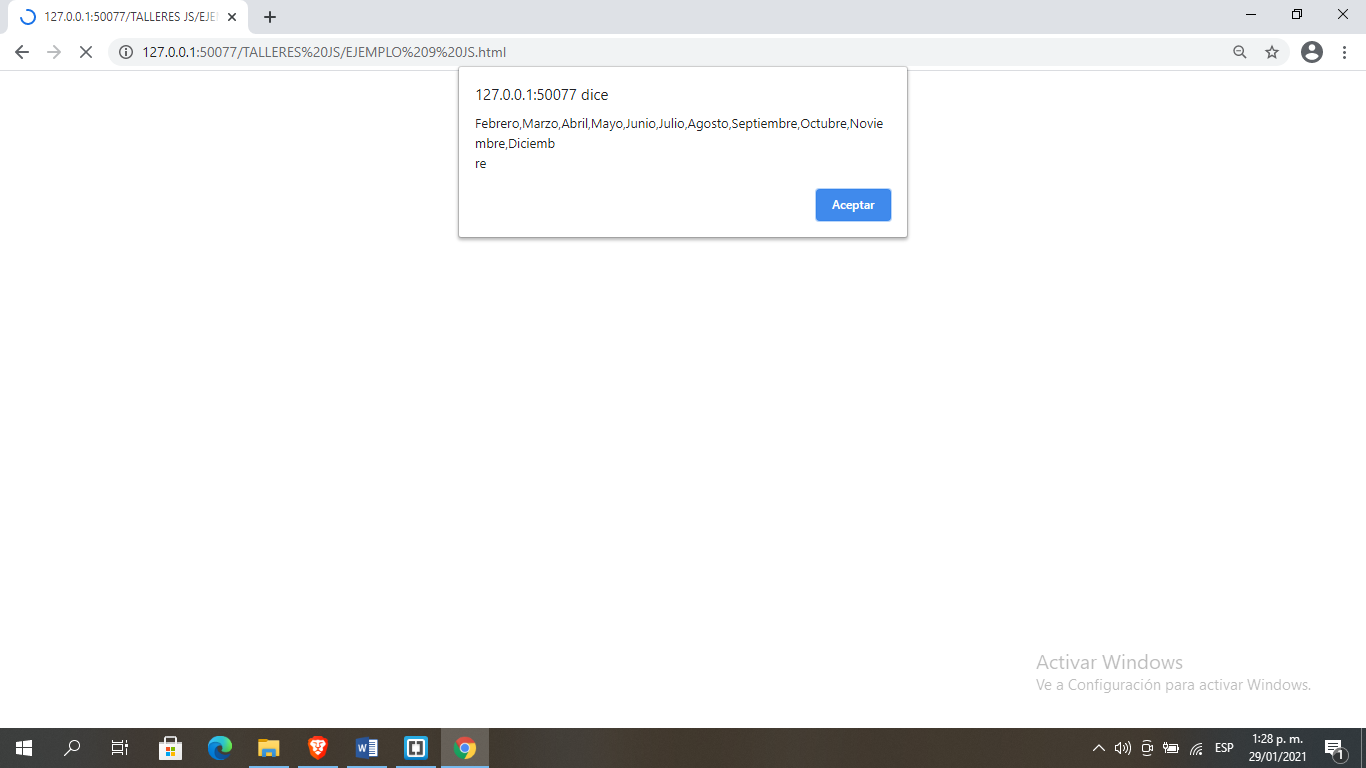
document.getElementById("texto").innerHTML=mes\_list.**shift()**;

var lista\_nueva =mes\_list

//

alert(lista\_nueva);

</script>



En este ejemplo el método shift( ) se utiliza para eliminar del array el elemento (Enero) al principio del mismo.

* + 1. **Método UNSHIFT()**

Método que se utiliza agregar un componente a un arreglo tipo vector (al principio), a su vez nos da la longitud del nuevo array.

Ejemplo:

<p id="texto"></p>

<script type="text/javascript">

var mes\_list =['Febrero','Marzo','Abril','Mayo','Junio','Julio','Agosto','Septiembre','Octubre','Noviembre','Diciembre'];

document.getElementById("texto").innerHTML=mes\_list.**unshift('Enero');**

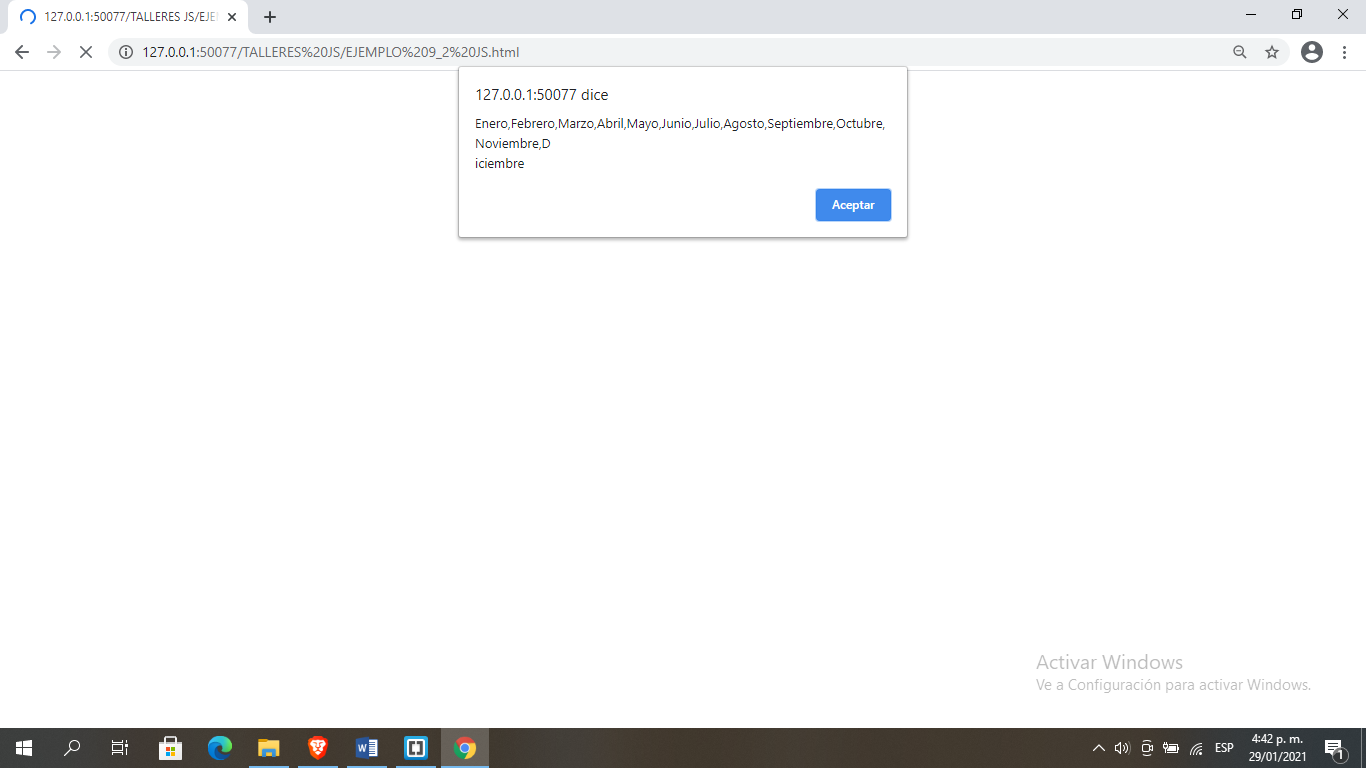
var lista\_nueva =mes\_list

//

alert(lista\_nueva);

</script>

En este ejemplo el método unshift( ) se utiliza para agregar en array el elemento (Enero) al principio del mismo.



* + 1. **Método MATH**

Se utilizan para realizar operaciones matemáticas con valores numéricos.

**Math.round () …**Redondea número al entero más próximo (a partir de 0.5 lo aproxima hacia el entero mayor, si es menor a 0.5 lo aproxima al entero menor).

**Math.abs () …** Genera el valor absoluto o positivo del número.

**Math.floor () …**Redondea número al entero más próximo (por debajo).

**Math.ceil () …**Redondea número al entero más próximo (por arriba).

**Math.random () …**Genera un numero aleatorio de 0 a 1.

**Ejemplo**

**Math.round(3.67); ….** //resultado 4

**Math.round(3.47); ….** //resultado 3

**Math.abs(-4.5); ….** //resultado 4.5

**FDEH**

* Estudiante para afianzar conocimientos adquiridos debes de colocar el método correcto

var sem\_list =['lunes','martes','Miercoles’,'jueves','viernes','sabado'];

document.getElementById("texto").innerHTML=sem\_list.\_\_\_\_**(**'Domingo'**);**

* Utilice HTML DOM para cambiar el valor del campo de entrada.

<input type="text" id="dato" value="Dar clic">

<script>

document.getElementById("dato")\_\_\_\_\_\_\_\_\_\_= “Javascript”;

</script>

## **Estructuras secuenciales de un programa**

* + 1. **Captura de datos en Javascript**

**Instrucción prompt()** : este comando me permite generar un cuadro dotado de los elementos propicios para permitir el ingreso de información por teclado; su sintaxis es:

**prompt(‘Mensaje de entrada’,’Valor por defecto’);**

Ejemplo:

En este ejemplo se declara las tres variables y luego se pide por teclado (**prompt**) los datos de nombre, precio y cantidad de un producto

var precio;

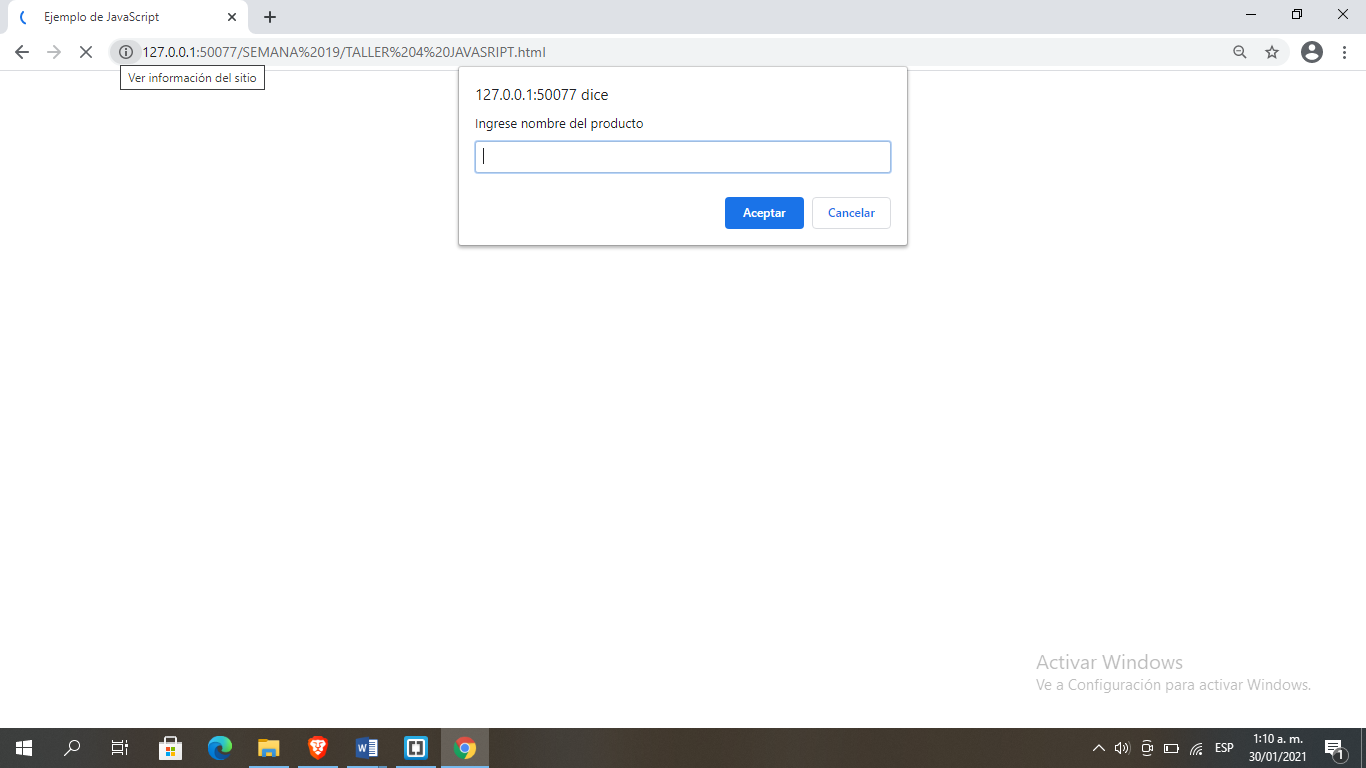
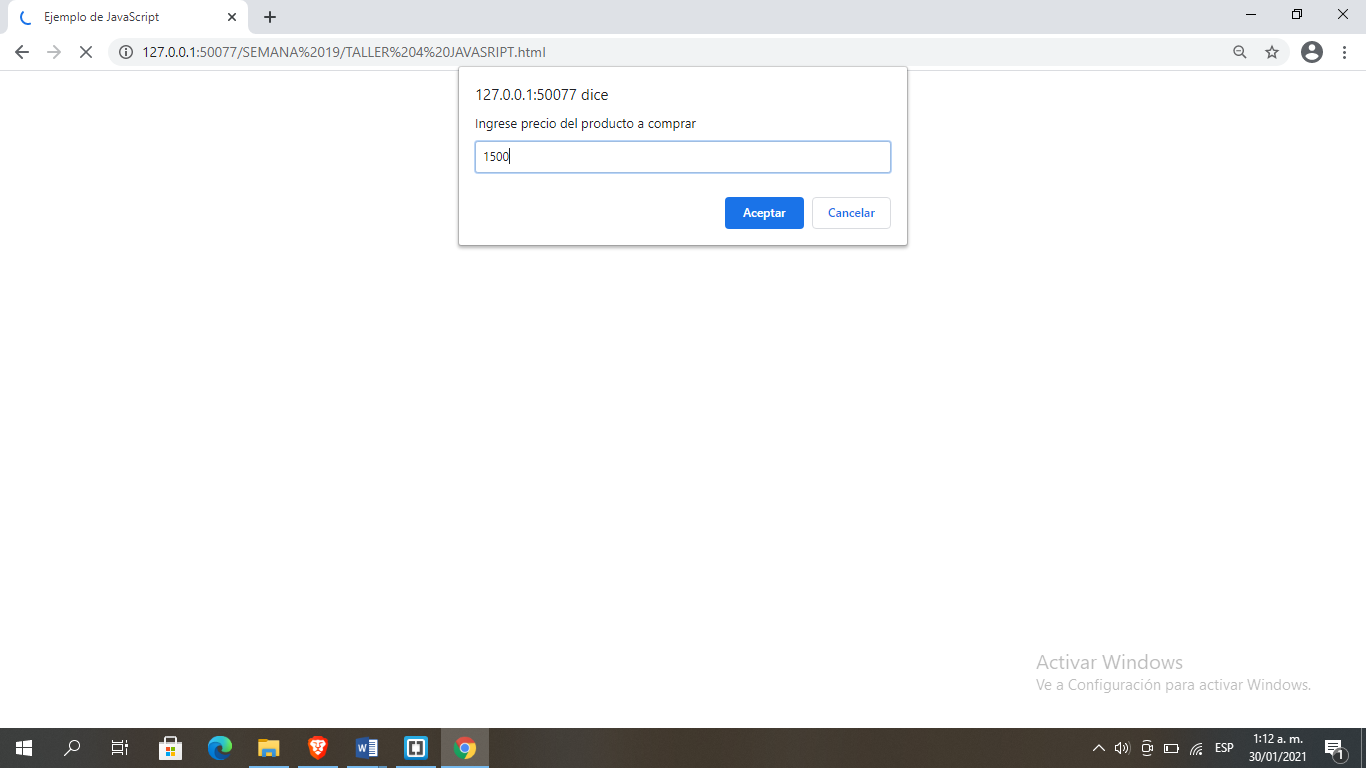
var cant;

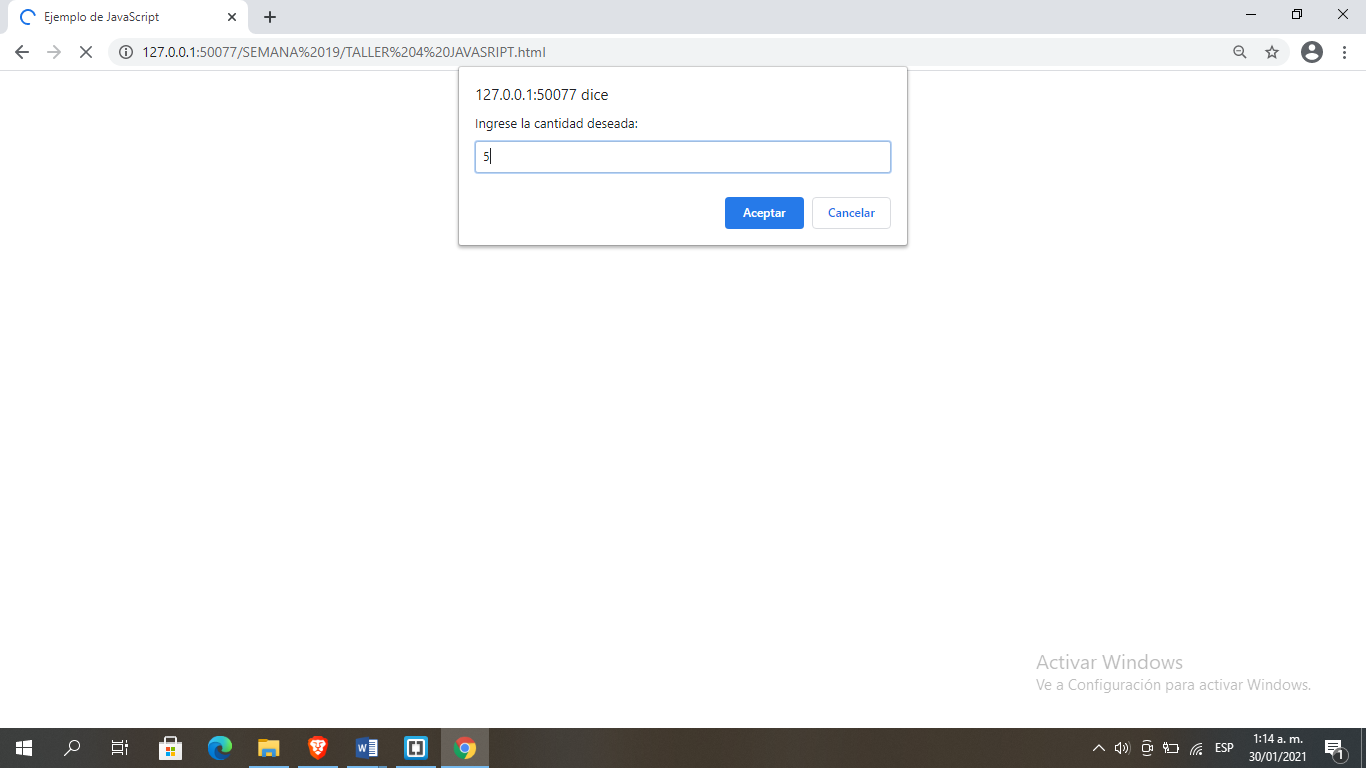
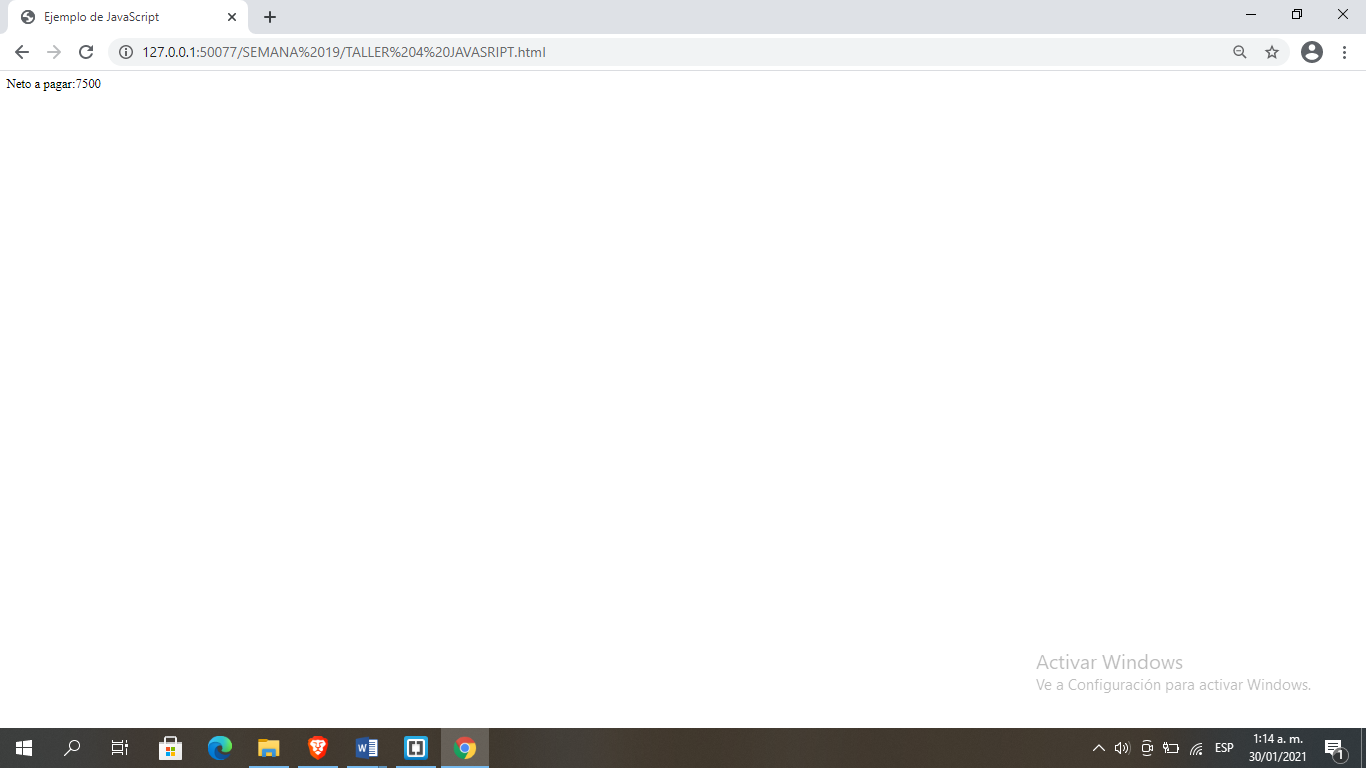
var nombrep;

nombrep=**prompt**('Ingrese nombre del producto','');

precio=**prompt**('Ingrese precio del producto a comprar','');

cant=**prompt**('Ingrese la cantidad deseada:','');

El comando anterior viene acompañado cuando se quiere que el resultado se muestre directamente en la página HTML; para ello utilizaremos el comando *document.write()* y su sintaxis es:

* + 1. **Salida de datos en Javascript**

**Instrucción document.write();**

Este comando nos permite mostrar los resultados de los procesos generados en la salida del navegador. Veamos un ejemplo que muestra las ventajas de dichos comandos, su sintaxis es:

**Document.write(‘mensaje’ + variable)**

<script>

En este ejemplo se declara las tres variables y luego se pide por teclado (prompt) los datos de nombre, precio y cantidad de un producto y al final se muestra en la salida del navegador, la instrucción **document**.**write** del valor a pagar.

var precio;

var cant;

var nombrep;

nombrep=prompt('Ingrese nombre del producto','');

precio=prompt('Ingrese precio del producto a comprar','');

cant=prompt('Ingrese la cantidad deseada:','');

var netop;

netop=parseInt(precio)\*parseInt(cantidad);

**document.write**('Neto a pagar:');

**document.write**(netop);

</script>

## **Ejemplos con instrucciones básicas**

**EJEMPLO 1**

**En los siguientes ejemplos utilizaremos variables diversas y la instrucción document.write**

<!DOCTYPE html>

<html>

<head>

<title>EJEMPLO 1 JAVASCRIPT</title>

<meta charset="UTF-8">

Se crean y asignan valores a las variables de tipo texto, numéricas y boolenas al final se muestran en la salida del navegador.

</head>

<body bgcolor=lightblue>

<script type="text/javascript">

var nombre='Pedro';

var edad=50;

var altura=1.75;

var soltero=false;

document.write(nombre);

document.write('<br>');

document.write(edad);

document.write('<br>');

document.write(altura);

document.write('<br>');

document.write(soltero);

</script>

</body>

</html>

**EJEMPLO 2**

**Desarrolle un script para generar el área de un triángulo.**

<!DOCTYPE html>

<html>

Se crean y asignan valores a las variables numéricas base y altura para generar el área del triángulo al final se muestran en la salida del navegador.

<head>

<title>Ejemplo 2 de instrucciones de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<script>

var base;

var altura;

base=prompt('Ingrese la base del triangulo:','');

altura=prompt('Ingrese altura del triangulo');

var areat;

areat=(base\*altura)/2;

document.write('El area del triangulo es:');

document.write(areat);

</script>

</body>

**</html>**

**EJEMPLO 3**

**Desarrolle un script que permita ingresar 4 valores distintos y genere una sumatoria y una productoria con esos números y muestre en pantalla los resultados respecticos.**

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo 3 de instrucciones de JavaScript </title>

<meta charset="UTF-8">

</head>

Se crean y asignan cuatro valores a cuatro variables numéricas se pide por teclado y se genera la sumatoria y la productoria de los números al final se muestran en la salida del navegador.

<body>

<script type="text/javascript">

var num1;

var num2;

var num3;

var num4;

num1=prompt('Ingrese primer valor:','');

num2=prompt('Ingrese segundo valor:','');

num3=prompt('Ingrese tercer valor:','');

num4=prompt('Ingrese cuarto valor:','');

var suma;

suma=parseInt(num1)+parseInt(num2)+parseInt(num3)+parseInt(num4);

var producto;

producto=parseInt(num1)\*parseInt(num2)\*parseInt(num3)\*parseInt(num4);

document.write('La suma de los cuatro valores es:');

document.write(suma);

document.write('<br>');

document.write('El producto de los cuatro valor es:');

document.write(producto);

</script>

</body>

</html>

**EJEMPLO 4**

**Desarrollar un script donde se pida por teclado los siguientes datos: nombre, precio y cantidad de un producto a facturar y mostrar por pantalla su valor antes de iva, su descuento 10%, su iva 16% y su neto a pagar**.

<!DOCTYPE html>

<html>

<head>

Se crean y asignan siete valores a siete variables se piden por teclado y se genera los datos de una factura subtotal, descuento, iva y el neto a pagar al final se muestran los resultados en la salida del navegador.

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<script>

var precio, cant, desc, iva, subt, netop;

var nombrep;

//

nombrep=prompt('Ingrese nombre del producto','');

precio=prompt('Ingrese precio del producto a comprar','');

cant=prompt('Ingrese la cantidad deseada:','');

//

subt=parseInt(precio)\*parseInt(cant);

desc=subt\*0.10;

iva=(subt-desc)\*0.16;

netop=subt-desc+iva;

//

document.write('SUBTOTAL:'+'<br>');

document.write(subt+'<br>');

document.write('Descuento:'+'<br>');

document.write(desc+'<br>');

document.write('Iva:'+'<br>');

document.write(iva+'<br>');

document.write('Neto a pagar:'+'<br>');

document.write(netop);

</script>

</body>

**</html>**

**FDEH**

Estudiante para afianzar este conocimiento debes de realizar los siguientes scripts:

* Sumar tres números que da el usuario
* Calcular el promedio de un estudiante el cual ingresa 3 notas
* Hallar el producto de 3 números y mostrar el resultado obtenido.
* Hallar el valor del día de un empleado que gana $8.000 por hora.
* Diseñe un scripts para determinar el área del cuadrado.
* Diseñe un scripts para determinar el área de un rectángulo.
* Diseñe un scripts que me permita calcular el porcentaje de hombres y de mujeres de un número cualquiera de personas (se debe pedir el número de personas, el número de hombres y el número de mujeres), enviar el porcentaje de hombres y de mujeres por pantalla.
* Diseñe un scripts que permita generar la velocidad de un vehículo teniendo en cuenta su espacio recorrido y su tiempo gastado.

## **Estructuras de control de flujo de un programa**

## **Transferencia de control condicional básica(if)**

Esta estructura de programación es muy en los diferentes lenguajes de programación ya que permite que un programa evalué una o varias condiciones y dependiendo del resultado de dicha evaluación el propio programa tome decisiones si el resultado de la misma es falso (false) o es verdadero (true).

Su sintaxis en JavaScript es:

**if (condición) {**

**Procesos por verdadero;**

**}**

**Else {**

**Procesos por falso;**

**}**

Si la condición analizada o evaluada se cumple o su resultado es true entonces se procede a realizar el o los procesos por verdadero que se encuentran dentro de las llaves {…}; si no se cumple se procede a realizar los procesos por falso que están limitados por las llaves después el programa sigue su flujo normal.

Veamos un ejemplo donde aplicamos el condicional básico:

<script>

var nombre;

var nota;

nombre=**prompt**('Ingrese nombre estudiante:','');

nota=parseInt(**prompt**('Ingrese su nota final:',''));

**if (nota>=3)**

{

document.write(nombre + ' tu materia es aprobada con un ' + nota);

}

**else**

{

document.write(nombre + ' tu materia es reprobada con un ' + nota);

}

</script>



* Complete y corrija la instrucción if para enviar una alert "Estudia Javascript" si x es mayor que y, de lo contrario, enviar un alert "Estudia PHP".

If ( \_\_\_\_\_\_){

\_\_\_\_\_\_\_\_\_\_

}\_\_\_\_\_{

\_\_\_\_\_\_\_\_\_\_

}

* Complete la instrucción if que ingresa dos valores al azar y se debe de realizar las comparaciones pertinentes.

**if ( ) {**

**alert(“El Valor1 No es mayor que el Valor2”);**

**}**

**if ( ) {**

**alert( “El Valor2 es positivo”)**

**}**

**if ( ) {**

**alert(“El Valor1 es negativo o diferente de cero”)**

**}**

**FDH**

## Transferencia de control condicional anidado (if/else)

En la vida real y en los procesos empresariales reales, lo normal es que se presente una condición y dependiendo del resultado de la evaluación de la misma se deban realizar varios procesos si el resultado es verdadero (true) o Y procesos si el resultado es falso (false); es entonces, cuando la condicional con esta estructura (if/else) se convierte en una herramienta de vital importancia.

Su sintaxis general es:

**if (condición) {**

**Procesos por verdadero;** **}**

**Else if (condición) {**

**Procesos por verdadero;**

**}**

**Else** **{**

**Procesos por falso;**

**}**

La sintaxis indica que si la condición evaluada es verdadera, se realizaran los procesos que están dentro del bloque if(); de lo contrario, ósea si la condición no se cumple o es falsa se realizaran lo procesos que se encuentran dentro del bloque else {} y se evalúa la siguiente condición ya en la última se evalúa la condición falsa o ultima comparación.

Veamos un ejemplo:

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<p> INGRESE NOTAS DE 1 A 5</p>

<script>

var nota1,nota2,nota3,prom;

nota1=parseInt(prompt('Ingrese nota n1:',''));

nota2=parseInt(prompt('Ingrese nota n2:',''));

nota3=parseInt(prompt('Ingrese nota n3:',''));

prom=(nota1+nota2+nota3)/3;

//

if (prom>=4)

{

document.write('Sobresaliente...'+ prom);

}

else if (prom>=3) {

document.write('Regular...' + prom);

}

else {

document.write('reprobado...'+prom);

}

</script>

</body>

</html>

determinando el valor del subsidio de transporte; de lo contrario se envía un mensaje determinando que no tiene derecho a dicho subsidio.

## Ciclos repetitivos o Bucles:

* + - 1. **Ciclo for():**

Cuando dentro de un programa de computador se requiere que uno o varios procesos se repitan varias veces, entonces es necesario recurrir a los ciclos repetitivos o bucles y déjeme decirle que esta situación se presenta muchas pero muchas veces en la vida real; Uno de los ciclos repetitivos que más se utiliza de pronto por su fácil comprensión es el for(), y sobre todo en procesos con arreglos o arrays. La sintaxis general es:

**for (inicialización; Condición; incremento) {**

**Procesos a repetir;**

**}**

Explicaremos brevemente cada uno de los argumentos encerrados entre paréntesis:

* Inicialización: este argumento establece el punto de partida el valor inicial de la variable contador, encargada de controlar las veces que el ciclo se debe repetir.
* Condición: es la condición o criterio que determina si se continua con el ciclo o se termina el mismo.
* Incremento: es la forma en la que se controla que la variable contadora cambie y se incremente para que dicho ciclo tenga un final; de lo contrario y si esto no ocurre crearíamos un ciclo infinito u también llamado un loop repetitivo.

Veamos un ejemplo:

En este ejemplo creamos un ciclo con la instrucción **for** para generar los números consecutivos desde del 1 al 10.

<script>

**var f;**

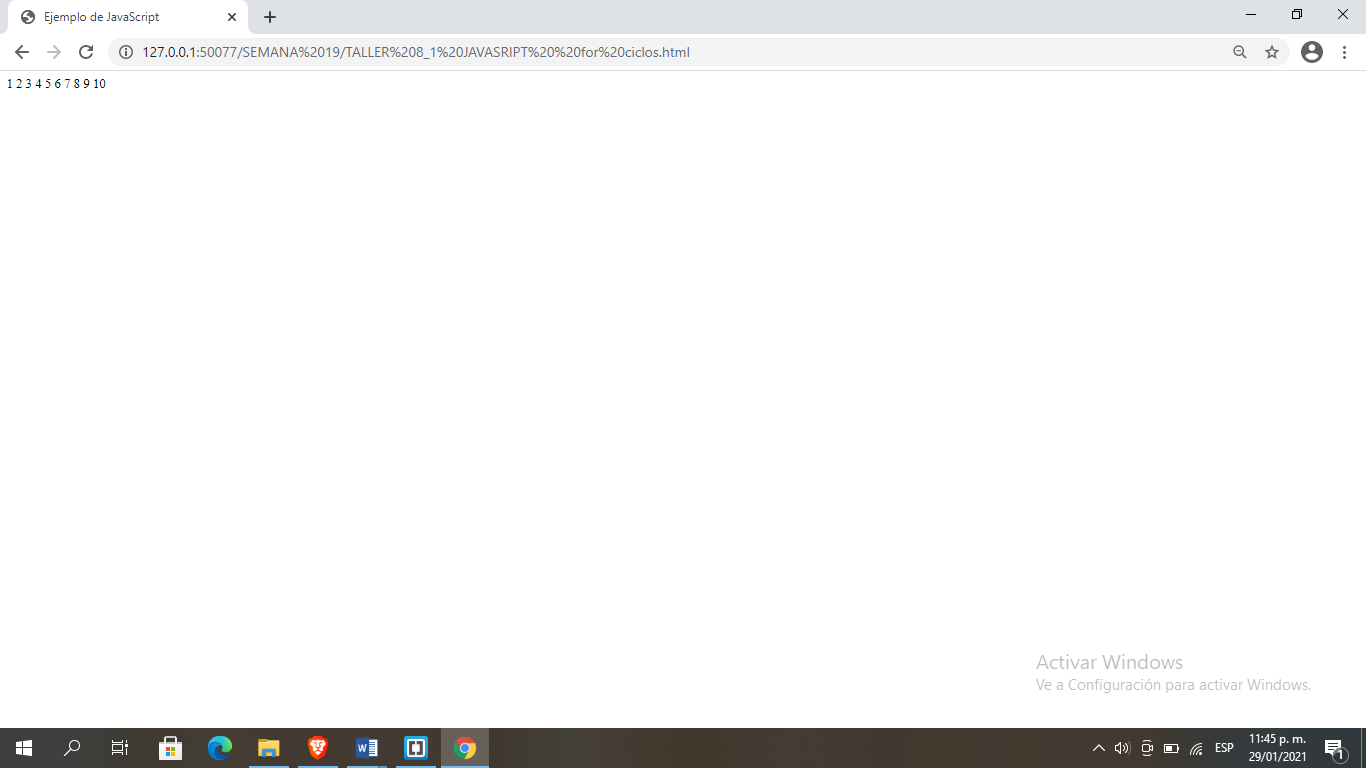
**for(f=1;f<=10;f++)**

**{**

**document.write(f+" ");**

**}**

</script>



Miremos otro ejemplo generar los números pares del 1 al 10:

<script>

En este ejemplo creamos un ciclo con la instrucción **for** unido a la instrucción **if** para generar los números pares desde del 2 al 10.

*Recuerda*: *se utiliza el símbolo == para comparar con 2.*

**var f;**

**for(f=1;f<=10;f++)**

**{**

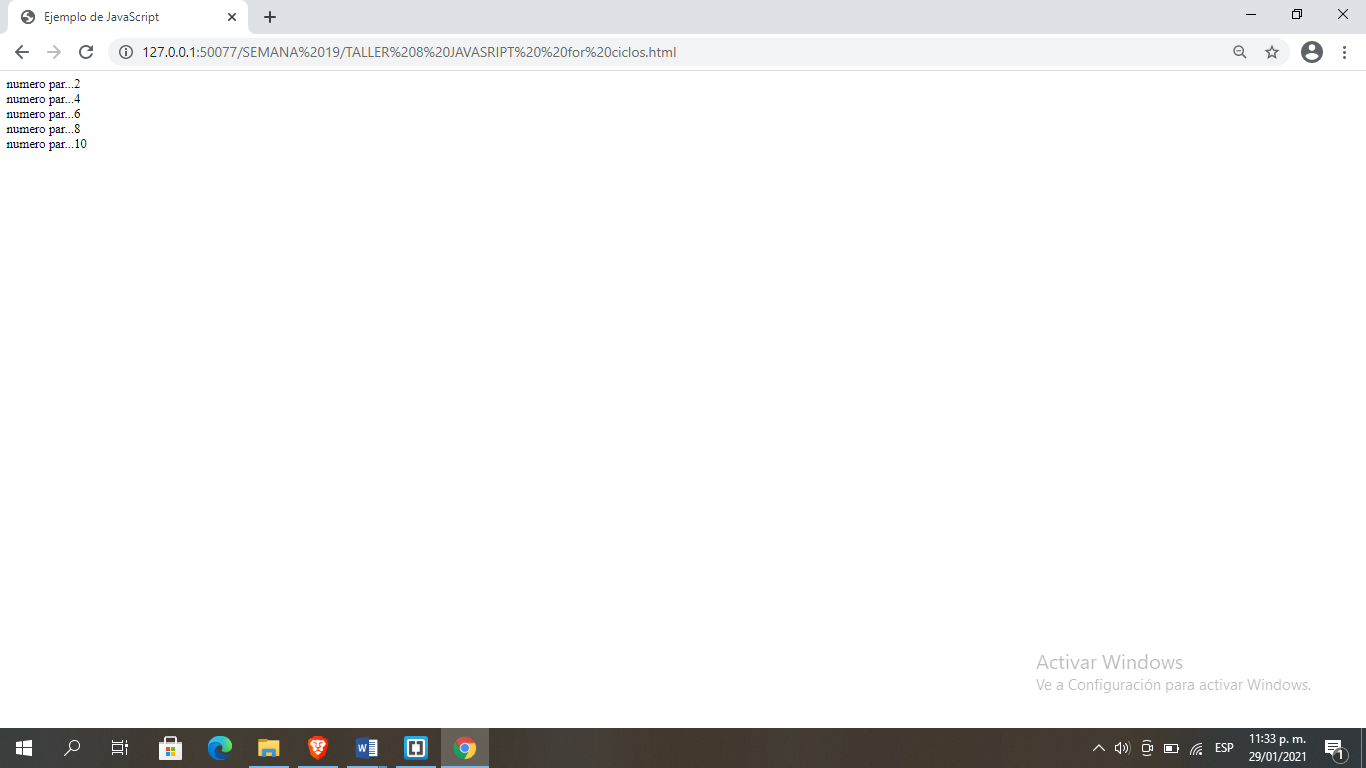
**if (f % 2 == 0){**

**document.write("numero par..."+f+"<br>");**

**}**

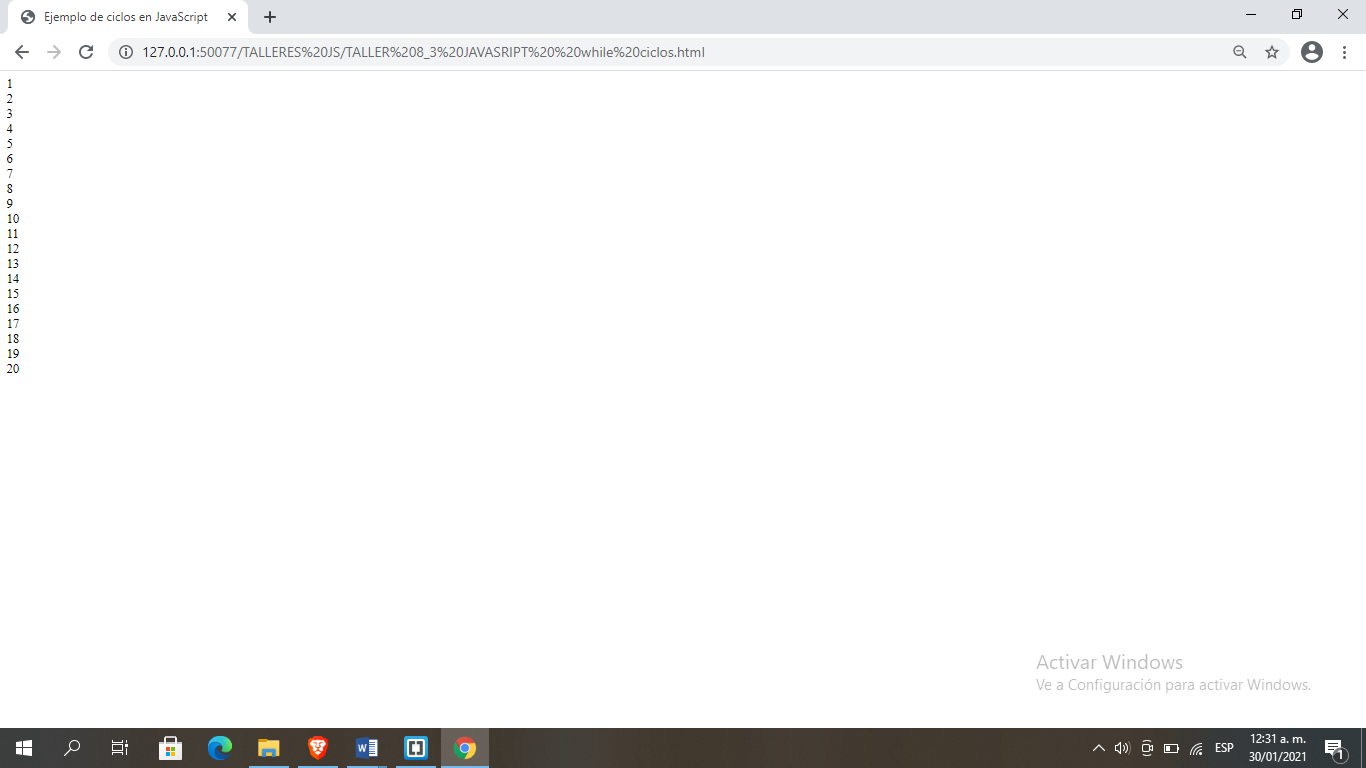
**}**

</script>



* + - 1. **Ciclos while**

En estos tipos de ciclos se evalúa primero la condición si es verdadera se realizan las instrucciones seguidas en caso que la condición sea falsa sale del ciclo y continúa el flujo del programa con las instrucciones siguientes a las llaves.

 Ejemplo:

En este ejemplo se generan los números consecutivos del 1 al 20.

**<script>**

**var num;**

**num=1;**

**while (num<=20)**

**{**

**document.write(num+"<br>");**

**num++;**

**}**

**</script>**

* + - 1. **Estructura switch:**

Cuando varios resultados dependen de una sola condición, en lugar de utilizar la sentencia if() es recomendable utilizar la sentencia switch cuya sintaxis es la siguiente:

**Switch(variable){**

**case valor1:**

**Procesos;**

**…….; break;**

**case valor2:**

**Procesos;**

**……; break;**

**case valorN:**

**Procesos;**

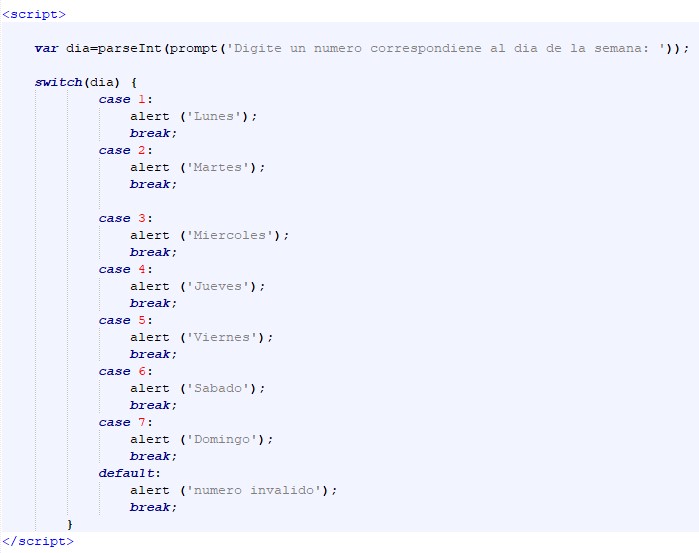
**…….; break;**

**default:**

**…….; break;**

**}**

Ejemplo:





* El estudiante creara un script que permita calcular el neto a pagar en una nómina de cinco empleados, la cual está determinada por las siguientes condiciones:
  + Al empleado se le paga un salario básico determinado por el contrato firmado.
  + Al empleado se le hace un descuento en pensión y salud correspondiente al 8% del salario básico devengado.
  + Al empleado se le cancela un auxilio de transporte según lo establecido por la ley si el salario de dicho empleado es menor o igual a dos salarios mínimos legales vigentes establecidos por la ley colombiana.
  + Se debe mostrar por pantalla el nombre del empleado y su neto a pagar.
* Crear un script que me permita crear un arreglo para guardar los meses del año y que luego los muestre uno a uno.

**FDH**

## Programación avanzada

Hasta ahora los scripts que se han desarrollado son de manejo básico y no requieren una estructura de programación muy compleja; en la vida real las aplicaciones en donde se involucra JavaScript requieren de mayor complejidad y es cuando el tema que a continuación expondremos, adquiere una importancia sustancial para que las aplicaciones a desarrollar funcionen de manera adecuada y eficiente.

Antes de adentrarnos en el concepto y ejemplos de Funciones creadas por el programador o desarrollador, veamos algunos temas que harán nuestra experiencia de programación más amena y fácil.

## Funciones Personalizadas

Cuando se requiere de ejecutar una instrucción o conjunto de instrucciones muchas veces dentro de una aplicación de mayor complejidad, es necesario recurrir a las funciones creadas por el programador o desarrollador para que de esta manera solo sea hacer un llamado a dicha función (invocarla) en el momento en que se desee utilizar.

Veamos la sintaxis de una función:

## function nombre de la función([parámetros]){ Instrucciones de la función;

**……;**

**}**

***Function*** es la palabra reservada que define la función.

Nombre de la función es el nombre que se dará a la función y con el cual se hará referencia al momento de ser llamada.

Parámetros son los valores de entrada que recibe la función para ser procesados; estos parámetros son opcionales y se deben colocar solo si la función los requiere.

Instrucciones de la función, son todos los procesos que debe realizar la función y van encerrados entre corchetes {}.

Para ejecutar la función tendremos que invocarla dentro de la página y en cualquier parte de esta; para estos procesos solo se debe escribir el nombre de la función seguido de los paréntesis.

NombreFuncion();

Veamos un ejemplo sencillo:

Se crea función bienvenida para que se pida por teclado el nombre del usuario, se asigna a una variable, luego llama a la función bienvenida y finalmente se ejecuta y envía mensaje de bienvenida pertinente por pantalla.

<script type="text/javascript">

**function bienvenida(nombre){**

**alert( 'hola como estas...' + nombre);**

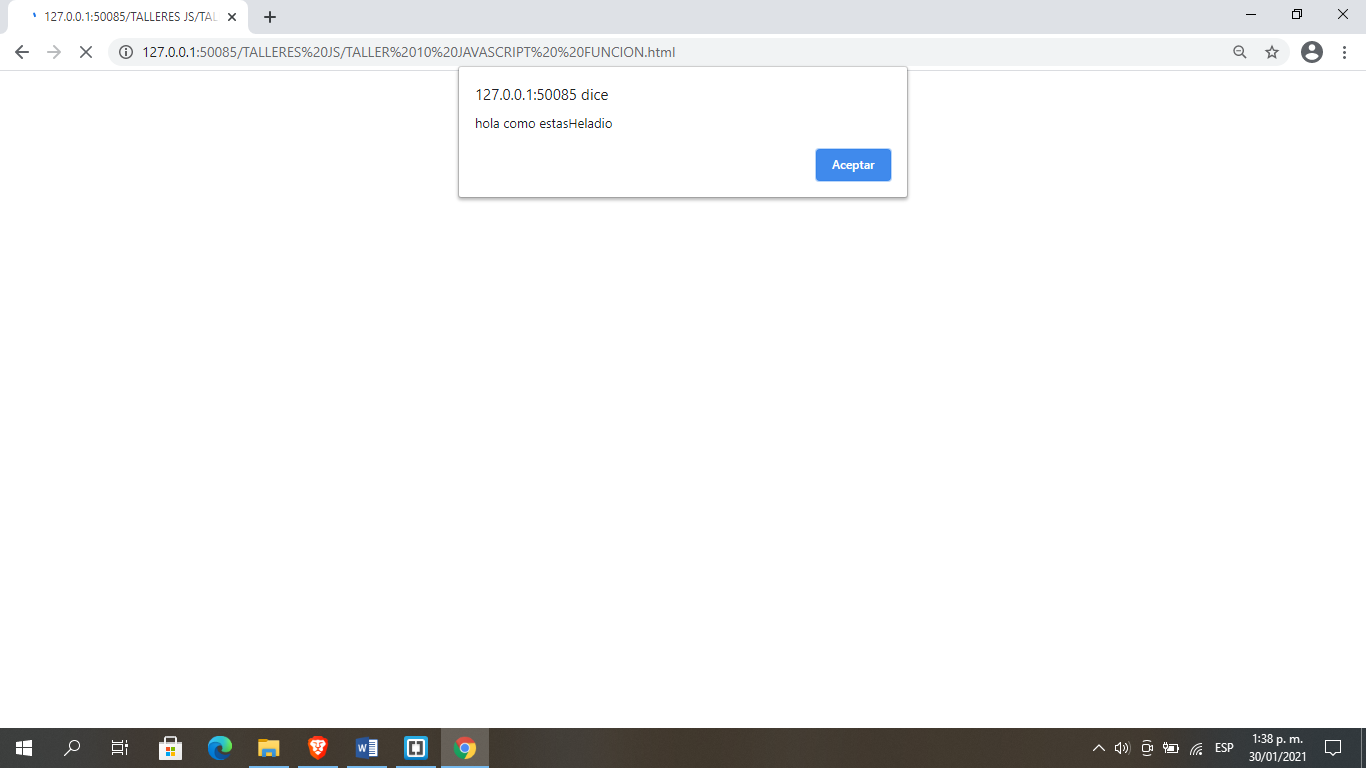
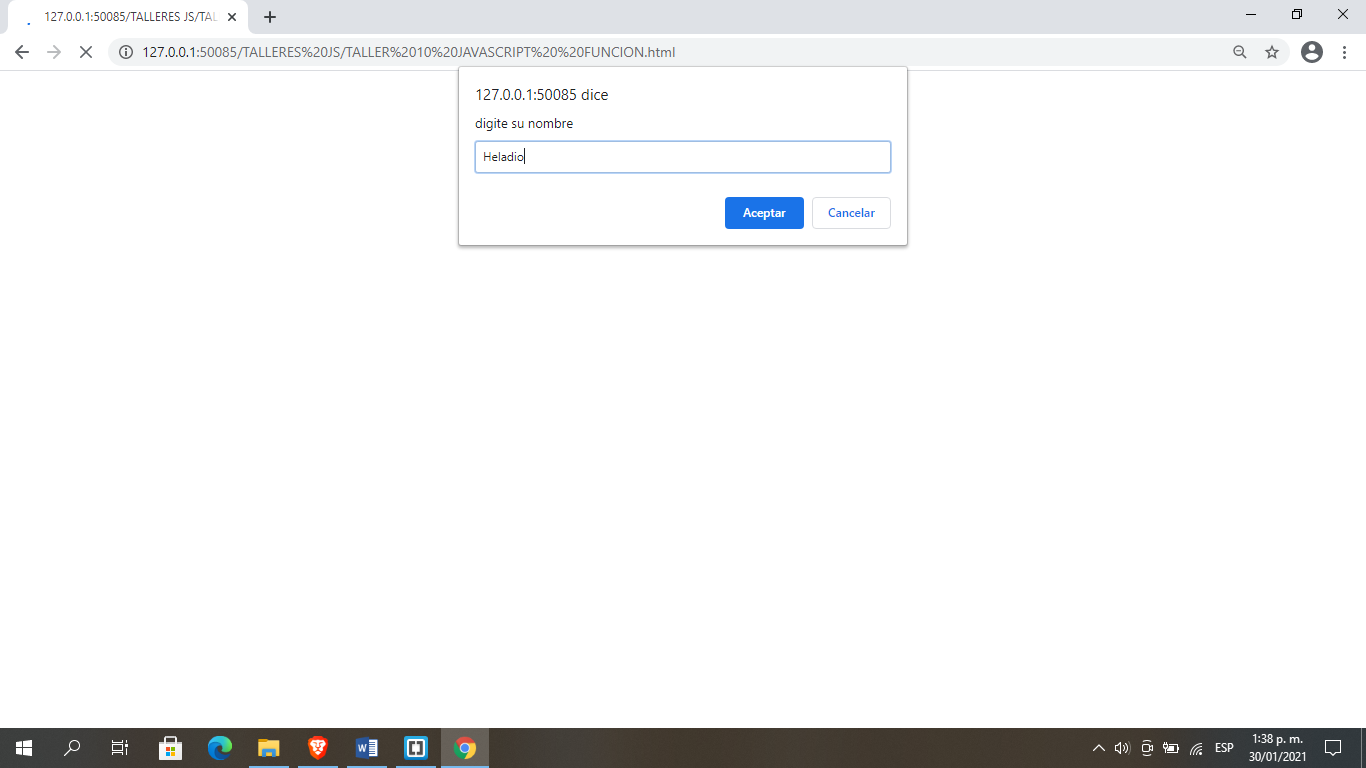
**}**

**var nombre=prompt( 'digite su nombre' , '');**

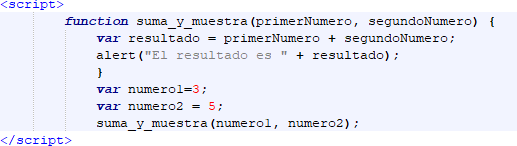
**bienvenida(nombre);**

</script>

RESULTADO FINAL



Aquí tenemos otro ejemplo, pero embebido en el cuerpo <body> de la página HTML:



Vamos a listar algunas características de los argumentos de una función:

* El número de argumentos que se envían a una función, se recomienda que sea el mismo que se ha indicado en la función, pero esto no es obligatorio y JavaScript no muestra error si el número de argumentos es diferente.
* El orden de los argumentos es de vital importancia y tiene que ser estricto, tal cual se establece en la función.
* Se puede utilizar el número que se desee de parámetros, aunque se recomienda que no se exagere en el mismo.
* No es obligatorio que los nombres de los argumentos que se definen en la función sean los mismos nombres que se le pasan.

FDEH

* Escribir el código de una función a la que se pasa como parámetro un número entero y devuelve como resultado una cadena de texto que indica si el número es par o impar. Mostrar por pantalla el resultado devuelto por la función.
* Crear una función que me permita realizar las operaciones básicas en una calculadora sencilla como la que se muestra en la imagen siguiente:

