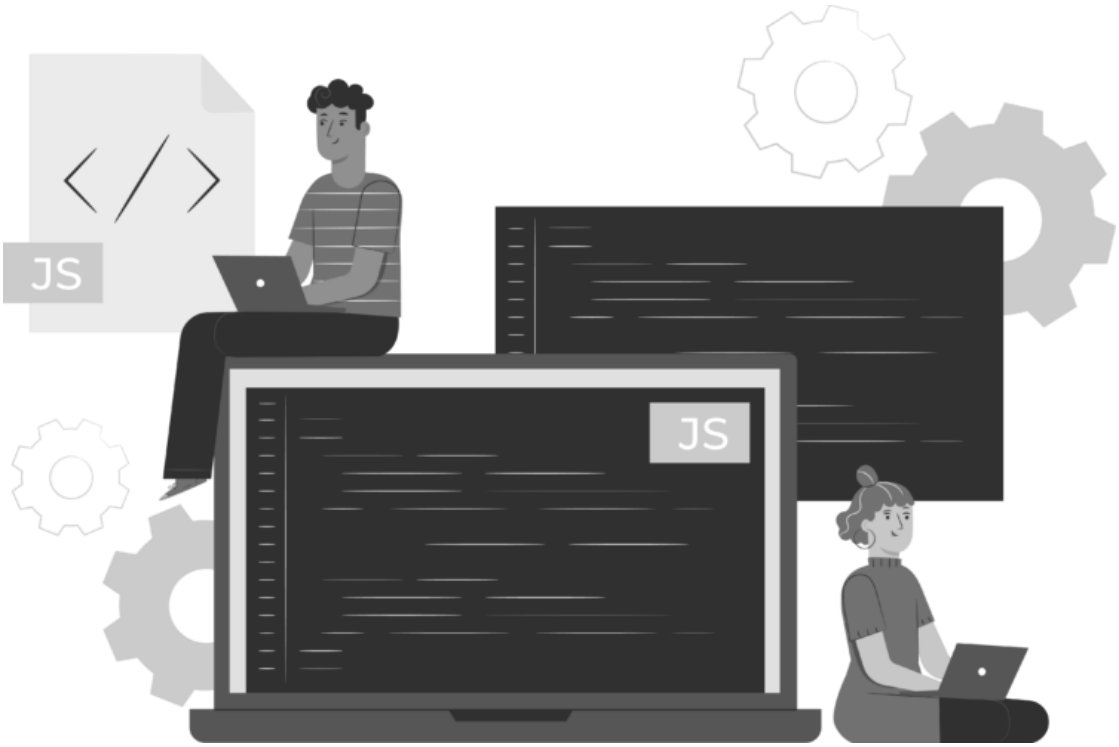
FUNDAMENTOS DE JAVASCRIPT

"Nada grande se ha hecho en el mundo sin una gran pasión.".

(Friedrich Hegel)

¿QUE ES JAVASCRIPT?

JavaScript (abreviado comúnmente JS), es un lenguaje de programación introducido en 1995 como una forma de añadir programas a las páginas webs en el navegador Netscape Navigator. Luego este lenguaje fue adoptado por el resto de navegadores webs. A finales de 1996 Netscape informó que traspasaba la responsabilidad de definir y estandarizar JavaScript a una organización internacional para el desarrollo de estándares en el sector de tecnologías de la información y comunicación denominada Ecma (European Computer Manufacturers Association). A partir de entonces, los estándares de JavaScript se rigen como ECMAScript. No solo JavaScript se basa el lenguaje ECMAScript, existen otros como JScript (Microsoft) y ActionScript (Adobe) que también lo hacen. Por eso, cuando en la web se habla de ECMAScript y JavaScript en realidad se está hablando de lo mismo. El primero es el estándar que define cómo debe funcionar el lenguaje, y el segundo es una implementación concreta de lo que indica la especificación estándar.
En el año 2015 aparece ECMAScript 6, también conocido como ECMAScript 2015 o ES6, esta nueva versión trae una gran cantidad de mejoras al lenguaje que nos ayudan al momento programar nuestras aplicaciones web. Desde entonces, los estándares ECMAScript están en ciclos de lanzamiento anuales. La última versión hasta el momento es ECMAScript 10(2019) lanzado justamente en el año 2019.



¿PARA QUE SIRVE JAVASCRIPT?

Gracias a JavaScript podremos interactuar con mapas, realizar animaciones en 3D, actualizar contenidos sin tener que refrescar o recargar la página, realizar juegos y responder a acciones de los usuarios mediante el uso de eventos. Este lenguaje debe ser aplicado a un documento HTML por lo que la interacción con HTML y CSS es fundamental. Entonces ahora resumiremos en pocas palabras lo que realiza cada una de estas tecnologías:

- HTML es un lenguaje de marcado donde definimos la estructura del documento (encabezados, párrafos, imágenes, videos, etc.) y el contenido o la información estática.
- CSS es un lenguaje de reglas que usamos para cambiar la presentación a nuestro contenido en HTML, por ejemplo, colocando colores de fondo, cambiando la tipografía, colocando el contenido en distintos márgenes, etc.
- JavaScript es un lenguaje de programación completo que permitirá añadir interactividad en el documento HTML.

Nota

No necesita JavaScript para crear un sitio web, pero sin JS la web sería solo páginas con contenidos estáticos de información; sin interactividad con el usuario, sin animaciones y con constantes recargas de páginas para obtener nueva información.

CONCEPTOS BASICOS DE JAVASCRIPT

Un programa es un conjunto de instrucciones escrito por un programador que realiza una determinada tarea sobre una computadora. En el caso de JavaScript esas líneas de instrucciones son interpretadas línea a línea por el Motor de JavaScript del navegador.
Estas instrucciones se denominan sentencias y deben tener un orden de ejecución. En el caso se JavaScript se separan por punto y coma (;).
JavaScript diferencia el código escrito en Mayúscula y el código escrito en Minúscula (es case sensitive). Por lo tanto, una palabra escrita en mayúscula no es lo mismo que esa misma palabra escrita en minúscula. Para introducir JavaScript en una página en HTML debe usarse la etiqueta <script>:

```
<script>

  // Código en JS.

</script>
```

Si el llamado al código JavaScript es externo entonces debe escribir la siguiente línea dentro de la etiqueta <head> del documento HTML.

```
<script src="nombre_js.js"></script>
```

Donde el atributo src indica la ruta completa para encontrar el archivo js. La extensión .js al final del nombre del archivo indica que contiene código JavaScript. Todos los archivos externos JavaScript tendrán esa extensión.
Así como en HTML y CSS es posible escribir comentarios, también lo puede hacer en JavaScript usando las dos barras inclinadas (//) o encerrados entre los caracteres /* y */ en el caso de que quiera comentarios en varias líneas.

Nota

No hay que confundir JavaScript con Java. Java es un lenguaje totalmente diferente y del cual JavaScript tomó su nombre por motivos comerciales, debido a que el lenguaje Java era ya un lenguaje muy popular en aquellos tiempos.

VARIABLES

Las variables son contenedores en los que se puede almacenar valores para que más adelante se pueda recuperar. Para declarar una variable se usa la palabra var seguida del nombre que le queremos dar a ese contenedor.

```
var autor;
```

La variable nombre almacenara cualquier valor que se le indique. Para luego almacenar un valor debemos usar el operador asignación (=).

```
autor='Isabel Allende';
```

De esta forma la variable autor contiene el valor Isabel Allende. Por otro lado, la información que puede almacenar una variable son:

- Números
- Cadenas de texto (Strings)
- Booleanos
- Arreglos (Arrays)

- Funciones
- Objetos
- Nulos (Null)
- Indefinidos (Undefined)

Veremos a continuación algunos ejemplos de estos tipos de datos:

- 12 (Es un dato tipo numérico), puede utilizar los operadores matemáticos para también devolver un dato tipo numérico, 12+5 (devolvería 17). También puede usar decimales, por ejemplo: 15.80.
- "Saludos gente linda", es un dato de tipo cadena de caracteres o String. Los strings van encerrados con comillas dobles o simples. Un error que sucede comúnmente es la interpretación de números y texto, es decir "10" o '10' no es lo mismo que 10.
- false, es un valor booleano significa falso, true por lo contrario es verdadero. Por ejemplo, la variable esAdulto puede ser false o true según la persona.
- Si declara una variable sin asignarle ningún valor su tipo será indefinido o undefined.
- var mascotas = ["perro", "gato", "loro"], es la declaración de un arreglo (array) un tipo que puede almacenar más de un valor.

Nota

JavaScript es un lenguaje tipado débil o dinámico, es decir, no es necesario indicar que tipos de valores puede almacenar una variable como en otros lenguajes como JAVA o C. El tipo será determinado automáticamente cuando el programa comience a ser procesado. Esto también significa que puedes tener la misma variable con diferentes tipos.

SCOPE

El scope (alcance) es el ámbito donde declaramos nuestras variables, por lo tanto, el scope es la lista de variables a las cuales tenemos acceso en un determinado momento. El scope puede ser de dos tipos: global o local. Una variable cuyo scope es global se puede acceder desde cualquier parte del código, una local solo desde el bloque que la contiene.
Un bloque de sentencias se utiliza para agrupar un conjunto de sentencias. Los bloques en JavaScript se delimitan mediante llaves, una de apertura ({), y otra de cierre (}).
En JavaScript se pueden declarar variables usando también la palabra reservada var, pero su uso es desaconsejable ya que se puede acceder a su valor desde cualquier parte del código del programa (uso global). Por otro lado, las variables definidas con let tienen ámbito de bloque (block scope). Esto significa que estas variables existen dentro del bloque donde son declaradas. Un buen principio de la programación nos indica que las variables deben declararse en el menor scope posible, de esta forma podemos evitar errores lógicos, por ejemplo, el asignar incorrectamente un valor a una variable, otra ventaja de hacerlo es que el código es más legible en la mayoría de los casos.

COMENTARIOS

Un comentario en un programa sirve para colocar una nota explicativa que el intérprete no ejecuta. Un comentario de una sola línea en JavaScript comienza con un par de barras diagonales (//). Un comentario de varias líneas de JavaScript comienza con una barra diagonal seguida de un asterisco (/*) y termina con los mismos elementos en orden inverso (*/).

CONSTANTES

A diferencia de una variable, una constante (declarada con la palabra reservada const). Contiene un valor y ese valor será inmutable y no podrá asignársele un nuevo valor más adelante. En el caso de intentar asignar un valor obtendremos el siguiente error: TypeError: Assignment to constant variable.

```
const nuestroPlaneta = “Tierra”;
```

Si en algún momento consideraríamos que nuestro Nuevo planeta es el planeta Marte. Daria un mensaje de Error.

```
nuestroPlaneta = “Marte”;  // TypeError: Assignment to constant variable.
```

INTERACCION CON EL USUARIO

Podemos hacer que sea el usuario el que le dé el valor a una variable. Para ello hacemos uso de la palabra reservada prompt() . Con esto, se abre una ventana que pide al usuario que introduzca un valor por pantalla.

```
let autor = prompt('Ingrese el Nombre');
```

En este caso luego de que el usuario ingrese el nombre este valor se almacenara dentro de la variable denominada autor.

OBJETOS

Los programas que hemos creado hasta ahora se han visto limitados por el hecho de que solo funcionaban con tipos de datos simples. Por ejemplo, vamos a suponer que queremos definir el uso de una tarjeta de crédito dentro de un programa. Si definiera número de tarjeta de crédito seria por ejemplo un tipo de dato simple como un String. Pero una tarjeta de crédito es más que un número, tiene un tipo (Visa Classic, Mastercard Gold, etc), un banco emisor, la fecha de emisión, etc.

Es decir, una tarjeta de crédito es un objeto con muchas propiedades que definen sus características. Estas propiedades pueden contener cualquier tipo de datos, incluidas números, strings y booleanos. También como veremos más adelante pueden contener métodos, que son funciones contenidas en un objeto. Para crear objetos, podemos hacerlo de varias maneras; nosotros usaremos la notación más extendida que es la notación literal donde entre llaves ({}) indicaremos las propiedades y su valores con la forma: propiedad:valor separador por comas. Por ejemplo, el siguiente código crea un objeto tarjeta de crédito:

```
let myCardCredit = {
    numero: '4111 1111 1111 1111',
    titular: 'Claudio E. Alonso',
    tipo: 'Visa Classic'
};
```

El objeto myCardCredit tiene tres propiedades número, titular y tipo cuyos valores correspondientes son: '4111 1111 1111 1111', 'Claudio E. Alonso' y 'Visa Classic'.
Para acceder a una propiedad de un objeto, se utiliza la notación punto ejemplo:

```
myCardCredit.numero;
```

FUNCIONES

En JavaScript las funciones son unos de los elementos mas importantes. Cuando desarrollamos suele pasar frecuentemente que hay muchas funcionalidades que se repiten dentro de nuestro programa. Las funciones permiten encapsular dicha funcionalidad para reutilizarla dentro del código. En su forma más básica una función tiene un nombre (funciones declarada o declarativa) y debe ser creada antes de su invocación.

```
function createAlarm()
{
        let sonido="Pling";
}
```

Esta es una simple función llamada createAlarm() dentro de ella encerrados entre llaves encontramos el código que ejecutara luego se ser invocada. Definir una función no la ejecuta. Definir una función simplemente la nombra y específica que hacer cuando la función es invocada Para ser invocada la función debe simplemente colocarse el nombre de la función en el lugar donde se ejecutará:

```
createAlarm();
```

No solamente podemos crear nuestras propias funciones, sino que también podemos usar las que nos provee el lenguaje. Ejemplo:

```
alert("Aquí estoy!");
```

Esta función nos permite mostrar un mensaje dentro de una ventana. El mensaje se define cuando se invoca a la función.
Cada función devuelve el valor undefined, a menos que se especifique lo contrario, para ello debemos indicar que valor queremos devolver después de ser ejecutada. Usando la instrucción return indicamos que valor devolver:

```
function createAlarm()
{
        let sonido="Pling";
        return "La alarma ha sido creada";
}
```

La función devuelve un string “La alarma ha sido creada” cuando es invocada. Por ejemplo, la podremos invocar así:

```
alert(createAlarm());
```

En este caso usamos la invocación de una función dentro de una función que nos provee el lenguaje. Por la tanto el mensaje a mostrar en la ventana es: La alarma ha sido creada.
Otro punto fundamental para tener en cuenta es que las funciones con frecuencia toman parámetros (valores) que se colocan dentro de los paréntesis y son útiles para que la función pueda realizar correctamente dicha funcionalidad. En el caso de que los valores sean mas de uno se separan con comas (,).

```
function createAlarm(dia,hora)
{
        let sonido="Pling";
        return "Alarma creada!, para el dia "+dia+" hora "+hora;
}
```

Nota

Los términos parámetros y argumentos a menudo se usan indistintamente. Sin embargo, son diferentes. Los parámetros se utilizan al declarar la función. Por otro lado, los argumentos son valores que la función recibe de cada parámetro en el momento en que se llama a la función

En este caso la función createAlarm tiene dos parametros, uno denominada día, otro denominado hora. Ahora escribiremos el código para invocar a la función anteriormente realizada.

```
<!doctype html>
<html lang="es">
<head>
        <meta charset="UTF-8" />
        <title>Document</title>
        <script src="alarma.js"></script>
</head>
<body>
<script>
        alert(createAlarm("Lunes", "07:00:00"));
</script>
</body>
</html>
```

FUNCIONES ANONIMAS

Un segundo tipo de función son las llamadas funciones anónimas donde no se especifica explícitamente el nombre para la función.

```
let nombreMascota=function() {
        let nombre = prompt('Ingrese el Nombre de su Mascota');
        alert(nombre);
};
```

En este caso guardamos a la función en una variable llamada nombreMascota (El valor de la variable es la función). Para invocarla necesitamos llamar al nombre de la variable.

```
nombreMascota();
```

Esta función, a diferencia de la anterior, es que no se carga en memoria hasta que no se ejecuta la línea de asignación (es decir se crean en tiempo de ejecución). Normalmente estas funciones son usadas sólo una vez en nuestro código.

METODOS EN LOS OBJETOS

Anteriormente vimos que los objetos en JavaScript son grupos de pares clave-valor. Los valores pueden consistir en propiedades, pero también de métodos. Los métodos son las tareas que puede realizar un objeto. Veremos cómo podemos definir un objeto en JavaScript:

```
let soldierRome = {
    name:'cesar',
    weapon: 'Gladius',
    fight: function() {
        return this.name+ " ataca con su: "+ this.weapon;
    }
};

alert(soldierRome.fight());
```

En este caso construimos un objeto Soldado Romano que usara su arma en un ataque. Para ello armaremos un método que nos permitirá a nuestro soldado luchar. La palabra this usada en nuestro método se refiere al objeto actual en el que se está escribiendo el código, por lo que en este caso this es igual a Soldado Romano.

OBJETOS INTRINSECOS

JavaScript define algunos objetos de forma nativa, por lo que pueden ser utilizados directamente por las aplicaciones sin tener que crearlos. Los objetos Boolean, Number, String aparte de almacenar los mismos valores de los tipos de datos primitivos añaden propiedades y métodos para manipular sus valores. Otros como los objetos Array y Date tienen una gran cantidad de propiedades y métodos. Para poder usarlos se declaran con la palabra reservada new y el nombre del objeto. Por ejemplo, supongamos que quiero obtener la fecha y la hora actual generado por mi Sistema Operativo:

```
let fecha = new Date();
```

Definimos una variable denominada fecha. Esta variable contiene un objeto de tipo Date que lo que hace es almacenar la fecha y la hora. Como fecha es un objeto podemos usar todas las propiedades y métodos que provee este objeto.
Por ejemplo si usamos el método getDay() este nos devolverá el día actual. También puede conseguir el año actual usando getFullYear().

Por ejemplo:

```
let fecha = new Date();
alert(fecha.getFullYear());
```

¿QUE ES UNA API?

El termino API viene del inglés Application Programming Interface, que en español seria Interfaz de Programación de Aplicaciones esto es un conjunto de funciones y métodos que permite a los desarrolladores crear programas sin tener que escribir códigos desde cero. En términos de desarrollo es como una capa de abstracción que representa la capacidad de comunicación entre distintos componentes de software. El concepto de API y su utilización es bastante anterior a la popularización de internet. Por ejemplo, existen apis para conectarse a un sistema operativo (WinAPI) o apis para comunicarse a una base de datos. Con el uso de internet se vio la posibilidad de que el software y las aplicaciones que podían ser interconectados a través de una API no tenían por qué estar juntos, sino que podían residir en sistemas remotos y simplemente tenían que poder comunicarse a través de internet, al fin y al cabo, lo que estaban haciendo era intercambiar información.

Nota

Por ejemplo, cuando un cliente compra entradas a través de la página web de un cine e introduce la información de su tarjeta de crédito, la web usa una API para enviar dicha información de forma remota a otro programa que verifica si los datos bancarios son correctos. Una vez que se confirma el pago, la aplicación remota envía la información al sitio web del cine y le da un OK, para que esta página emita las entradas.

Estos servicios de interacción entre sistemas a través de internet se denominaron servicios web. Para que estos servicios pudieran entenderse entre sí se necesitaron estándares, y uno de los más extendidos se denomina SOAP (Simple Object Access Protocol) y está basado en el lenguaje XML. Por otro lado, aprovechando el protocolo HTTP surgió el estándar REST. Las API REST permiten la utilización de servicios web disponibles a través de internet utilizando el protocolo HTTP usando JSON siendo mucho más simple que usar SOAP. Actualmente cientos de empresas usan API REST (Por ejemplo, Facebook y Twitter).
Un ejemplo claro de esto lo usa la empresa Fastway, dedicada al transporte de paquetes, y su API REST. La API tiene dos funciones: permite calcular el coste de envío de un paquete y permite conocer su estado de envío y entrega. Las empresas que utilizan los servicios de Fastway seguramente tendrán sus sitios web y/o aplicaciones móviles. La API de Fastway les permite dar un mejor servicio al incorporar en sus propios servicios online la posibilidad de informar a sus clientes el coste de los envíos y conocer en qué situación se encuentran estos. Es decir, los clientes de las empresas que usan de los Fastway no necesitan acceder al sitio web de Fastway para obtener información de sus envíos.

Para manipular los datos. HTTP nos otorga los métodos con los cuales podemos operar: GET (para consultar datos), POST (para crear datos), PUT (para modificar datos) y DELETE (para eliminar datos). Por ejemplo, para listar los productos debemos escribir la siguiente URL: get/productos (lista todos los productos) o get/productos/3333 (lista el producto cuyo código es 3333).

Nota

Otro ejemplo con APIs es la de Mercado Libre que se puede obtener en tiempo real los datos de los artículos publicados en Mercado Libre en su sitio web. Ejemplo de esto usando su api REST es https://api.mercadolibre.com/sites/MLA/search?q=dvd

MODELO DOM

El Modelo de Objetos del Documento (DOM) es un API para documentos HTML y XML. Este provee una representación estructural del documento, permitiéndole modificar su contenido y presentación visual mediante el uso de un lenguaje de scripting tal como JavaScript.
El DOM permite un acceso a la estructura de una página HTML mediante el mapeo de los elementos de esta página en un árbol de nodos. Cada elemento se convierte en un nodo y cada porción de texto en un nodo de texto. El nodo más importante es el nodo raíz (el padre de todos los elementos) llamado document.

Ejemplo:

```
<p>Esto es un párrafo que contiene <a href="#">un link </a> </p>
```

Como puede verse un elemento a se encuentra localizado dentro de un elemento p del HTML, convirtiéndose en un nodo hijo, o simplemente hijo del nodo p, de manera similar p es el nodo padre.
Afortunadamente, Javascript permite acceder a cada uno de los elementos de una página utilizando tan sólo algunos métodos y propiedades. Si desea encontrar de manera rápida y fácil un elemento puede usar el método getElementById. El mismo permite un acceso inmediato a cualquier elemento tan sólo conociendo el valor de su atributo id. Véase el siguiente ejemplo:

```
<p>
    <a id="contacto" href="contactos.html">Contáctenos</a>
</p>
```

Puede usarse el atributo id del elemento a para acceder al mismo:

```
let elementoContacto = document.getElementById("contacto");
```

El método getElementById es adecuado para operar sobre un elemento en específico, sin embargo, en ocasiones se necesita trabajar sobre un grupo de elementos por lo que en este caso puede utilizarse el método getElementsByTagName. Este retorna todos los elementos de un mismo tipo. Veamos este ejemplo:

```
<ul>
      <li><a href="somos.html">¿Quiénes Somos?</a></li>
      <li><a href="productos.html">Productos</a></li>
      <li><a href="servicios.html">Servicios</a></li>
      <li><a href="contactos.html">Contáctenos</a></li>
</ul>
```

Puede obtenerse todos los hipervínculos de la siguiente manera:

```
let enlaces= document.getElementsByTagName("a");
```

El método document.write() permite mostrar algo en una página web, por ejemplo podríamos hacer algo parecido a esto:

```
document.write("Notas en la página");
```

Escribirá el texto Notas en la página. Otro método que se usa bastante es el método innerHTML, el mismo permite establecer o devolver el contenido HTML de un elemento. Por ejemplo, supongamos que tenemos el siguiente menú de opciones:

```
<ul>
      <li><a id="somos" href="somos.html">¿Quiénes Somos?</a></li>
      <li><a href="productos.html">Productos</a></li>
      <li><a href="servicios.html">Servicios</a></li>
      <li><a href="contactos.html">Contáctenos</a></li>
</ul>
```

Vamos a cambiar el texto del primer <li> pero dinámicamente usando JavaScript, para ello podemos usar el método innerHTML:

```
let elementoSomos = document.getElementById("somos");
elementoSomos.innerHTML = "Nosotros";
```

Nota

Manipular el DOM es tan importante en el desarrollo web que han aparecido librerías específicamente destinadas a facilitar esta tarea, como jQuery.

EVENTOS

En la programación tradicional, las aplicaciones se ejecutan secuencialmente de principio a fin para producir resultados. Sin embargo, mediante el uso de eventos un programa puede esperar un suceso y luego de haberse producido podemos brindar respuestas mediante la ejecución de código.

Nota

Un ejemplo de evento en un juego de un partido de futbol es cuando el árbitro pita el silbato cuando finaliza el partido. De esta forma los jugadores son avisados de la finalización del encuentro.

Uno de los ejemplos mas utilizado es cuando realizamos un click sobre un botón de una aplicación o cuando recargamos una página. La sintaxis es la siguiente:

```
element.addEventListener('eventType', eventHandler);
```

Donde element es el elemento que espera el suceso y eventType es un tipo de evento y el eventHandler es la función que se ejecutará cuando el eventType se dispare. Ahora pondremos el siguiente ejemplo:

```
let btn = document.querySelector('p');
btn.addEventListener('click', updateName);
```

Para este caso tenemos una variable btn que contiene un párrafo, luego este elemento queda a la espera de un evento click , cuando el usuario realice un click se ejecutará el código que se encuentra dentro de la función updateName.